PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Inteligência Artificial e Aprendizado de Máquina

Rômulo Ponciano da Silva Freitas

EXPLICABILIDADE COMO APOIO PARA DETECÇÃO DE FRAUDE EM TRANSAÇÕES FINANCEIRAS POR MEIO DE INTELIGÊNCIA ARTIFICIAL

Rio de Janeiro, Brasil
Abril de 2022

Rômulo Ponciano da Silva Freitas

EXPLICABILIDADE COMO APOIO PARA DETECÇÃO DE FRAUDE EM TRANSAÇÕES FINANCEIRAS POR MEIO DE INTELIGÊNCIA ARTIFICIAL

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Inteligência Artificial e Aprendizado de Máquina, como requisito parcial à obtenção do título de *Especialista*.

Rio de Janeiro, Brasil

Abril de 2022

SUMÁRIO

1. Introdução

Desde o início dos bancos, existem pessoas que tentam roubar as informações dos clientes para retirar o dinheiro das contas. No início era comum o roubo de identidade para se passar por outra pessoa a partir de informações pessoais e falsificação de assinaturas. Entretanto, com o avanço da tecnologia se tornou cada vez mais comum o uso de métodos para invadir as contas através de vazamento de dados e acesso a senhas óbvias [6, 8].

Atualmente, entre os diferentes tipos de fraudes envolvendo instituições financeiras como, por exemplo, roubo de identidade e clonagem de cartão de crédito, existe um relacionado ao acesso à conta bancária da pessoa. Quando o criminoso consegue acesso à conta, ele retira o dinheiro da vítima via saques presenciais ou transações online [1].

O crescente aumento no uso de celulares e sistemas bancários para realização de transações entre contas provocou um aumento no número de roubo de contas e vazamento de dados de clientes de instituições financeiras por todo o mundo [2]. No Brasil, com a chegada do sistema de transações Pix, ainda houve um aumento de 39% no número de sequestros relâmpagos somente em São Paulo [3]. Estes problemas forçaram até mesmo o Banco Central a adotar medidas em conjunto com as instituições financeiras do país [4].

Para solucionar estes problemas, a área de Machine Learning evoluiu e realizou diversos estudos e projetos em grandes bancos com o intuito de treinar modelos preditivos e utilizá-los em ambiente de produção. Entretanto, esta estratégia se depara com diversos problemas. Entre eles, um dos mais conhecidos é o uso de datasets extremamente desbalanceados [5, 7].

Alinhado a isso, a incerteza presente nos modelos preditivos cria a necessidade de utilizar os modelos preditivos como ferramentas de apoio à

decisão onde a decisão final fica a cargo de alguma pessoa. Porém, muitos modelos preditivos criam formas de realizar suas predições através de soluções caixa-pretas [9, 10]. Ou seja, soluções que não conseguimos inferir facilmente.

Por isso, este projeto busca avaliar e comparar o uso de modelos preditivos caixa-preta com aqueles que possuem maior capacidade de interpretação em seus resultados. Ao utilizar modelos facilmente explicáveis, podemos receber até *feedbacks* de usuários destas ferramentas que nos auxiliem a melhorar o modelo, seja reclassificando registros para uma futura atualização ou descartando regras que podem levar o modelo a predições preconceituosas [9].

2. Contextualização

Hoje em dia, além de medidas de segurança preventivas e a conscientização com os usuários, as instituições financeiras tentam identificar este tipo de fraude através do treinamento de modelos preditivos classificatórios treinados por meio de algoritmos de aprendizado de máquina, visto que todo mundo está suscetível a um sequestro como estes. Contudo, como existe um número muito maior de transações financeiras válidas do que aquelas identificadas como fraude, os dados coletados acabam divididos de forma totalmente desbalanceada [5, 6].

O problema de dados financeiros tão desbalanceados é, inclusive, um dos fatores que mais afeta negativamente o treinamento e uso de modelos treinados para este fim [7]. Além disso, esses modelos acabam sendo utilizados como ferramentas de apoio à decisão pois sabe-se que são imperfeitos e acarretam em predições equivocadas [8].

Por esse motivo, é muito importante que as predições feitas pelos modelos possam ser interpretadas de alguma forma por aqueles que utilizarão o apoio, mesmo que seja uma predição diretamente enviada ao cliente como, por exemplo, “identificamos uma transação suspeita na sua conta. Faça ação x para confirmar ou y para bloquear”.

Essa necessidade por interpretação na predição dos modelos gerou

um crescente interesse em uma subárea chamada Explicabilidade em Inteligência Artificial (XAI) [9]. Com estudos nessa área, foi possível separar modelos entre aqueles chamados de explicáveis e modelos caixa-preta. O primeiro diz respeito a modelos que fornecem uma interpretação de suas predições, seja por regras de associação, probabilidade ou mesmo sequência lógica dos dados. Já o segundo diz respeito a modelos onde não podemos inferir com facilidade os motivos por trás de suas predições como, por exemplo, modelos de aprendizado profundo [10, 11].

3. Descrição do Problema e da Solução Proposta

Atualmente muitos estudos e práticas focam na detecção de fraude em transações financeiras e outros estudos focam na explicabilidade de modelos preditivos treinados por meio de algoritmos de aprendizado de máquina [12, 13]. Entretanto, como descrito anteriormente, é importante ter a capacidade de interpretar as predições neste caso. Não só para melhorar o próprio aprendizado do modelo em uma futura atualização com dados mais precisos, como também aumentar o entendimento do problema perante o usuário que irá se deparar com o resultado da predição.

Por isso, o objetivo deste trabalho de conclusão de curso é demonstrar as dificuldades de se trabalhar com dados desbalanceados, o treinamento de diversos modelos para os fins de predição de fraude em transações financeiras e comparar os resultados entre os modelos caixa-preta e aqueles com maior facilidade de explicabilidade de suas predições. Dessa forma, o trabalho segue o fluxo demonstrado na Figura 1.

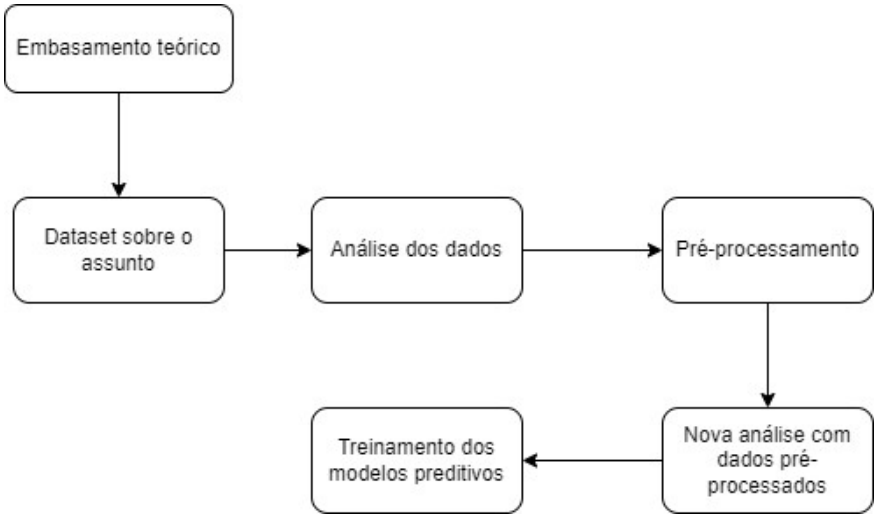


Figura 1. Fluxo do projeto

Nesta Figura 1 pode-se notar que o primeiro passo após o embasamento teórico sobre o problema foi a busca por uma base de dados que representasse transações financeiras contendo transações válidas e fraudulentas. Com a base de dados em mãos, foi preciso fazer uma análise destes dados que, por sua vez, revelou o pré processamento necessário. Após o pré processamento, houve uma nova análise em cima dos dados processados e, por fim, foi realizado o treinamento dos modelos preditivos e uma análise final sobre os resultados obtidos.

4. Coleta de Dados

Para o objetivo deste trabalho, é importante que o dataset seja desbalanceado e esteja no contexto do problema: fraudes em transações financeiras. Neste aspecto, não é o intuito do trabalho generalizar o resultado para o mundo todo ou um país específico, mas sim apresentar os resultados de uma análise e treinamento em um dataset desbalanceado e com o intuito de comparar diferentes algoritmos de aprendizado de máquina em relação a

sua qualidade e explicabilidade.

Por esse motivo, a base de dados foi buscada e encontrada no site Kaggle[9.1]. Este site é um dos repositórios de dados mais famosos para a disponibilização e desafios relacionados a Big Data envolvendo Data Science e Machine Learning. O dataset foi disponibilizado pelo colaborador Vardhan Siramdasu sob licença de dado aberto publicamente e acessado no dia 25/02/2022 através do próprio Kaggle[9.2].

4.1. Características dos Dados

A base de dados possui mais de 6 milhões de registros (6362620) e 10 colunas. Cada registro representa uma transação em um espaço de tempo onde as linhas se encontram ordenadas cronologicamente dentro de um período de 30 dias. As colunas estão dispostas na base de acordo com as seguintes características:

step (Inteiro): representa a hora cronológica em que a transação ocorreu no período dos 30 dias. Por exemplo, step 1 significa que ocorreu na primeira hora dos 30 dias; step 73 significa que ocorreu na 73ª hora (1ª hora do terceiro dia). Neste dataset, uma mesma hora pode conter 0 ou n transações.

type (String): representa o tipo daquela transação. Os tipos possíveis são: CASH-IN, CASH-OUT, DEBIT, PAYMENT ou TRANSFER.

amount (Double): quantidade financeira da transação, independente do tipo

nameOrig (String): código do cliente de origem por onde parte a transação

oldbalanceOrg (Double): quantidade financeira na conta de origem antes da transação.

newbalanceOrig (Double): nova quantidade financeira na conta de origem após a transação ser efetivada

nameDest (String): código do cliente de destino para onde a transação se destina

oldbalanceDest (Double): quantidade financeira na conta de destino antes da transação

newbalanceDest (Double): quantidade financeira na conta de destino após a transação ser efetivada

isFraud (Boolean): classe alvo do dataset. Valor 1 significa que aquela transação é uma fraude, enquanto o valor 0 significa que não é uma fraude.

isFlaggedFraud (Boolean): atributo criado para simular um algoritmo que marca uma transação como possível fraude se, e somente se, a transação for em um valor maior que 200 mil.

A Figura 2 demonstra as primeiras 5 linhas presentes no dataset.

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlagge
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0	
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0	
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1	
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1	
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0	

Figura 2. Cinco primeiras linhas presentes no dataset

Além disso, o dataset possui uma regra sobre sua limitação e já especificada no problema: clientes que possuam código com inicial de M são clientes que não possuem informações financeiras sobre a conta de destino daquela transação (*oldbalanceDest* e *newbalanceDest* sempre zerados)

4.2 Distribuição dos Atributos

Nesta seção é apresentada a distribuição e características estatísticas de cada um dos atributos presentes no dataset.

4.2.1 step

Podemos ver pela Figura 3 que a distribuição do atributo não é uma distribuição normal. Com uma média de 243.4 e um desvio padrão de 142.33, tem-se uma grande quantidade de transações concentradas entre a 1ª e 8ª hora. Após uma queda no número de transações, tem-se novos picos

entre a 110ª e 410ª hora.

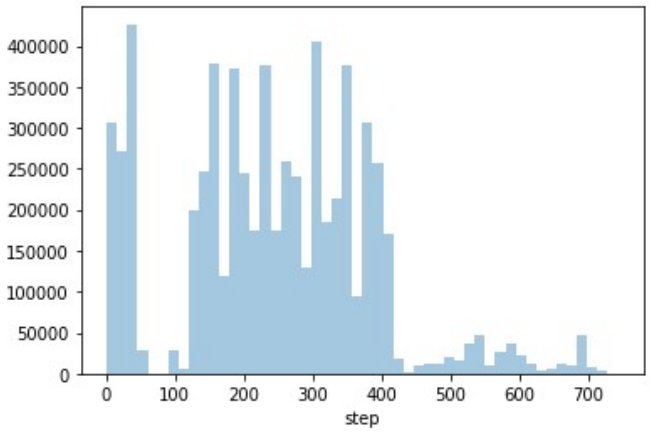


Figura 3. Distribuição do atributo step

Os valores do Step variam entre 1 e 743, onde 25% destes valores se encontram até o valor 156, 50% até o valor 239 e 75% até o valor 335. Reforçando, mais uma vez, que a maior parte das transações aconteceram antes da hora 335.

4.2.2 type

Podemos ver pela Figura 4 que existem muito mais transações dos tipos PAYMENT e CASH_OUT do que transações do tipo TRANSFER e DEBIT. Já o tipo CASH_IN se encontra com uma quantidade média em torno das maiores.

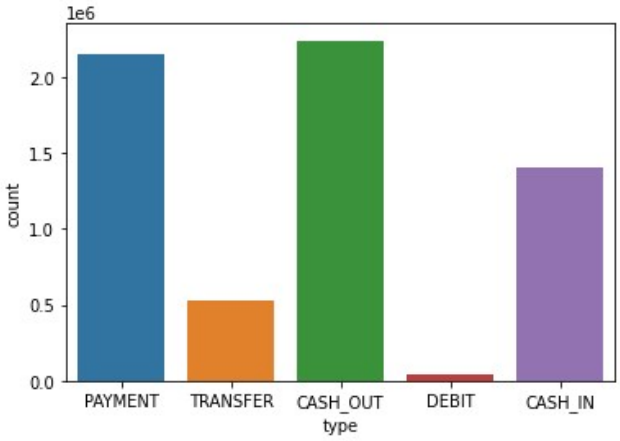


Figura 4. Quantidade de cada tipo de transação no dataset

Os valores absolutos de cada tipo são: 2237500 CASH_OUT, 2151495 PAYMENT, 1399284 CASH_IN, 532909 TRANSFER e 41432 DEBIT.

4.2.3 atributos financeiros

Os atributos referentes a valores financeiros foram sumarizados nesta seção. Pode-se notar na Figura 5 que os valores representam números muito variados e espaçados, desde o valor zero até dezenas de milhões.

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFlaggedFra
count	6362620.00	6362620.00	6362620.00	6.362620e+06	6.362620e+06	636262
mean	179861.90	833883.10	855113.67	1.100702e+06	1.224996e+06	
std	603858.23	2888242.67	2924048.50	3.399180e+06	3.674129e+06	
min	0.00	0.00	0.00	0.000000e+00	0.000000e+00	
25%	13389.57	0.00	0.00	0.000000e+00	0.000000e+00	
50%	74871.94	14208.00	0.00	1.327057e+05	2.146614e+05	
75%	208721.48	107315.18	144258.41	9.430367e+05	1.111909e+06	
max	92445516.64	59585040.37	49585040.37	3.560159e+08	3.561793e+08	

Figura 5. Tabela de distribuição dos atributos financeiros

É importante notar também a última coluna da Figura 5. Essa coluna representa as métricas do atributo marcado pelo algoritmo automático que marcava como possível fraude uma transação com movimento maior ou igual a 200 mil.

4.2.4 atributo classe

Com uma quantidade absoluta de 6354407 transações válidas e apenas 8213, é fácil notar o quanto este dataset se encontra desbalanceado. Apenas 0.129% das transações são fraudes.

5. Processamento/Tratamento de Dados

Nesta seção é descrito como os dados foram processados e tratados, e quais ferramentas foram utilizadas para este processo.

5.1 Linguagem de programação e bibliotecas

Para este projeto foi adotada a linguagem de programação Python

[9.4]. Esta é uma linguagem amplamente adotada pela comunidade quando o assunto é Data Science ou Machine Learning [14]. Devido a essa grande adoção, a linguagem se tornou bem sólida para este problema em conjunto com uma variedade de bibliotecas para apoiar o desenvolvimento.

Todo o processamento foi feito através de um Jupyter Notebook. Este framework permite a utilização da linguagem Python em conjunto com bibliotecas que podem ser importadas e utilizadas entre marcações de texto. O motivo da escolha do Jupyter Notebook é para facilitar a explicação e o raciocínio por trás de cada etapa desde a análise inicial dos dados, passando pelo processamento até chegar no treinamento dos modelos.

As bibliotecas utilizadas para a execução deste projeto foram: ***numpy*** (processamento de arrays e matriz), ***pandas*** (processamento de dados de forma eficiente, importações e transformações de dataframes), ***matplotlib.pyplot*** (inserção de gráficos), ***sklearn.metrics*** (cálculo e exibição de métricas para algoritmos de aprendizado de máquina), ***sklearn.model_selection.train_test_split*** (separação de dados em conjuntos de treino e teste), ***sklearn.metrics.confusion_matrix*** (cálculo e exibição de matriz de confusão para analise de modelos preditivos), ***sklearn.metrics.classification_report*** (cálculo e exibição de relatórios para avaliação de modelos preditivos), ***seaborn*** (inserção de gráficos), ***joblib*** (salvar e carregar modelos preditivos), ***sklearn.manifold.TSNE*** (aplicação do algoritmo TSNE), ***matplotlib.patches.mpatches*** (inserção de gráficos de dispersão), ***sklearn.tree*** (aplicação do algoritmo árvore de decisão), ***sklearn.tree.export_graphviz*** (exportação de árvora de decisão montada a partir de um modelo), ***six.StringIO*** (exportação de imagem), ***IPython.display.Image*** (exibição de imagem), ***pydotplus*** (exportação e exibição de imagem), ***sklearn.neighbors.KNeighborsClassifier*** (aplicação do algoritmo de k-NN), ***sklearn.neural_network.MLPClassifier*** (aplicação do algoritmo MLP), ***sklearn.svm*** (aplicação do algoritmo SVM), ***sklearn.model_selection.GridSearchCV*** (busca de parâmetros ótimos para treinamento de modelos preditivos)

5.2 Problemas no dataset

A análise dos dados, resumida na seção 4.1, demonstrou que o dataset possui dois grandes problemas: (1) classe alvo completamente desbalanceada e (2) dados financeiros com muita variedade e espaçados. Além destes problemas, também pode-se elencar alguns que costumam ser comuns em datasets como, por exemplo, (3) presença de valores nulos ou registros que sem valores significados (caso dos clientes que começam com *M*).

5.2.1 Remoção de registros não significativos

Como trata-se de um dataset com milhões de linhas, pode-se remover dados problemáticos para os algoritmos sem grandes consequências. Entretanto, neste contexto é preciso se atentar para evitarmos ao máximo remover registros marcados como fraudes para não diminuir ainda mais a quantidade de registros do lado desbalanceado.

Após remoção dos registros nulos e registros com clientes que comecem com a letra M, o dataset passou a ter 4202912 de transações não fraudulentas e 8213 transações de fraude. Ou seja, não foi removida nenhuma linha relacionada a fraude.

5.2.2 Discretização dos atributos

Após apresentação na seção 4.2.3, ficou clara a necessidade de discretizar os atributos financeiros devido sua grande esparsidade e variação de valores. A discretização é um conceito que envolve transformar valores numéricos em uma categoria baseada em condições.

Neste contexto, foi aplicada uma discretização baseada nos quartis demonstrados na seção 4.2.3. Dessa forma, os valores foram distribuídos em categorias que definem se o valor está nos primeiros 25%, entre 25% e 50%, 50% e 75% ou acima de 75%.

Após a conversão, os atributos ficaram como dispostos na Figura 6. Nesta mesma figura também é importante notar que a discretização também foi aplicada nas colunas de String (*type*, *nameOrig* e *nameDest*) devido aos algoritmos de treinamento que não conseguem lidar com este tipo de dado.

Porém, nestes casos, a conversão foi mais simples: cada String

recebeu um número único. Assim, clientes receberam o mesmo *id* independente se estava na coluna *nameOrig* ou *nameDest*.

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
2	1	3	0	663412	1	0	439685	0	0	1	0
3	1	1	0	3659208	1	0	391696	0	0	1	0
9	1	2	0	3580202	2	2	282960	0	0	0	0
10	1	2	0	1958055	1	0	571261	0	0	0	0
15	1	1	2	4000507	1	0	417163	0	0	0	0

Figura 6. Cinco primeiras do dataset após discretização

5.3.3 Correlação dos atributos

Para identificar se outros atributos podem ser removidos ou se existe algum atributo com forte relação ao objetivo de predizer se é um registro de fraude ou não, utilizou-se o cálculo de Correlação de Pearson para esta análise. A Figura 7 apresenta o resultado.

	isFraud
step	0.039035
type	0.051358
amount	0.025079
nameOrig	-0.000582
oldbalanceOrg	0.044305
newbalanceOrig	-0.033464
nameDest	0.000495
oldbalanceDest	-0.040321
newbalanceDest	-0.023558
isFraud	1.000000
isFlaggedFraud	0.044095

Figura 7. Correção de Pearson entre os atributos e a classe

Sabe-se que, pela regra da Correlação de Pearson, quanto mais próximos o valor resultado for de 1 ou -1, maior é sua correlação positiva ou negativa, respectivamente. Assim sendo, nota-se que não existe um atributo único ou um conjunto pequeno de atributos que possua correlação suficiente para serem treinados separadamente dos outros. Entretanto, existem 2 atributos que possuem correlação próxima de zero: *nameOrig* e *nameDest*.

Pode-se até dizer que estes atributos possuem valor da correlação de zero, se arredondarmos para 3 casas decimais. Por isso, estes dois atributos também foram removidos do dataset.

5.3.4 Normalização dos atributos

Muitos algoritmos de Machine Learning realizam cálculos de distância entre os atributos. Porém, é necessário igualar estas magnitudes para que os algoritmos não considerem os valores de *step* com mais peso do que os outros atributos, pois *step* pode ir até a unidade de centena enquanto os atributos financeiros estão na casa de unidades.

Uma forma de realizar esta equalização é através da normalização de atributos. Existem diferentes técnicas para normalização de atributos e, neste projeto, optou-se pela normalização min-max devido sua praticidade e amplo uso [15].

5.3.5 Balanceamento da classe

Existem dois tipos básicos de balanceamento de dataset para classes desbalanceadas: simular e inserir registros semelhantes a classe desejada ou remover registros relacionados a classe com maior número [16, 17].

Para a realidade desta base e deste conceito, não podemos desconsiderar *outliers* pois estes podem justamente ser os casos de fraude. Além disso, simular mais casos randômicos pode criar registros de fraude associando comportamentos que não refletem uma possibilidade real e, assim, prejudicar o aprendizado em qualquer algoritmo.

Por esses motivos, optou-se por remover boa parte dos registros até que o desbalanceamento se torne menor. Ou seja, o objetivo não é remover completamente o desbalanceamento, mas sim reduzi-lo. Esta estratégia foi

adotada devido ao estudo de Krawczyk [18], que classificou o desbalanceamento de classes entre duas categorias: Altamente desbalanceada (1:1000+) e levemente desbalanceada (mínimo de 1:4).

Para diminuir o desbalanceamento para acima de 10%, foi preciso remover a maior parte do dataset. Após a remoção, o dataset ficou com 42029 transações não fraudulentas (83.65%) e 8213 transações fraudulentas (16.35%).

6. Experimentos e Análise com Machine Learning

Nesta seção é descrito os algoritmos que foram utilizados, as necessidades de mudanças no dataset em cada caso e as motivações para as escolhas destes treinamentos.

6.1 Avaliação dos Modelos

Devido a natureza desbalanceada do dataset, não é recomendável utilizar a métrica de acurácia como o valor a ser observado para avaliação. Isso ocorre pois o dataset se encontra desbalanceado e, portanto, é natural que a acurácia seja alta [16, 17].

Como o problema está diretamente ligado à necessidade de identificar corretamente quando uma transação é fraudulenta, deve-se olhar com mais atenção para os acertos relacionados a esta classe. Além disso, em caso de erros, esse contexto claramente se beneficia quando o modelo prediz uma falsa transação fraudulenta do que o aposto.

Por esse motivo, optou-se por avaliar o modelo através das métricas de Matriz de Confusão, que apresenta exatamente os acertos corretos (positivos e negativos) e os acertos falsos (positivos e negativos). Outra métrica diretamente ligada à Matriz de Confusão e também utilizada na avaliação deste projeto é o valor de *precision* e *recall*.

6.2 Separação de Dataset

Antes de realizar qualquer experimento com algoritmos de Machine

Learning, foi preciso separar o dataset em dados de treino e dados de teste. Este tipo de abordagem é extremamente recomendado para que o modelo possa ser avaliado de forma mais real. O modelo é treinado com a grande maioria dos dados e um conjunto separado de teste é reservado para avaliá-lo através das predições de registros que o modelo não usou durante seu treinamento anteriormente.

É importante destacar que esta abordagem é utilizada única e exclusivamente para avaliação do modelo. Ao ser definida a estratégia, todos os dados são utilizados para o treinamento de modelos que serão colocados em produção.

Para separar o dataset, foi utilizado o módulo *train_test_split* da biblioteca *sklearn*, como mostra a Figura 8. Com este método foi possível não só separar os dados escolhendo a porcentagem desejada, como também utilizar a estratégia de particionamento igualitário entre as classes. Ou seja, ao separar 20% dos dados para teste, o algoritmo garante que manterá a mesma proporção das classes no dataset.

```
def split_data(dataframe):
    y = dataframe['isFraud']
    X = dataframe.drop('isFraud', axis=1, inplace=False)
    return train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
```

Figura 8. Método criado para separação do dataset

6.3 Estudo sobre redução de dimensionalidade

Antes de aplicar qualquer dado aos algoritmos, foi feita uma análise sobre a redução de dimensionalidade dos atributos. Ao reduzir a dimensão de 9 atributos para 1, por exemplo, teria ganho não só no processo de treinamento como também uma aplicação maior de algoritmos para serem utilizados e avaliados em conjunto com os dados de dimensionalidade original.

O estudo foi feito através do algoritmo de Incorporação de Vizinhos Estocásticos t-Distribuídos (t-NSE). Este algoritmo é um método estocástico

para reduzir a dimensionalidade de dados e visualizar o resultado.

Ao aplicar o algoritmo através da biblioteca *sklearn*, obteve-se o resultado apresentado na Figura 9. Com esta figura, nota-se que não houve um padrão claro para que o dataset com dimensionalidade reduzida fosse aplicado em outros algoritmos.

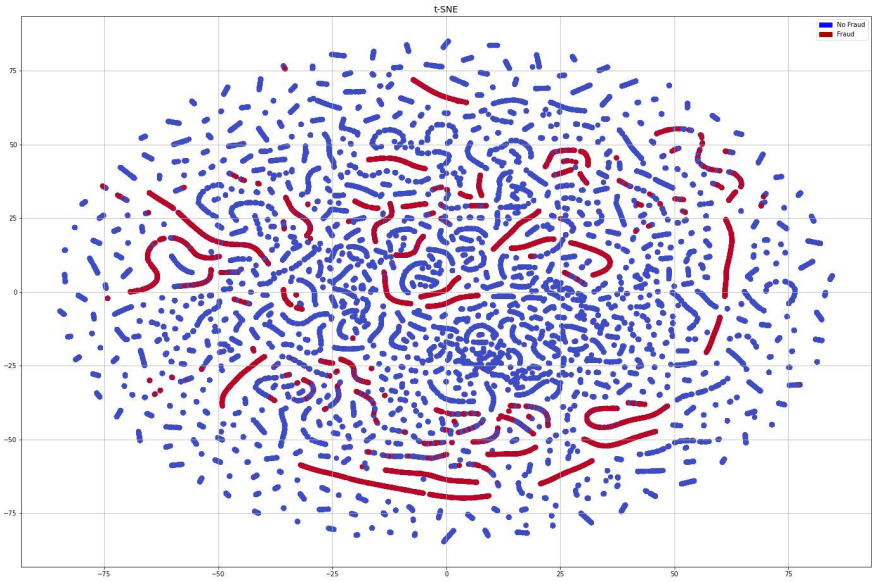


Figura 9. Demonstração do resultado após t-NSE

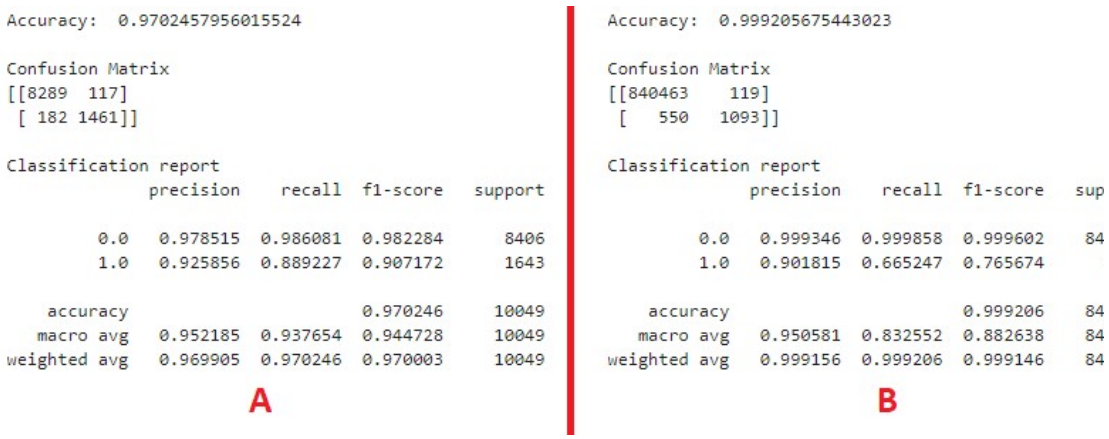
É importante destacar que o algoritmo foi aplicado com diferentes parâmetros e este foi o resultado mais claro obtido, como mostra o notebook completo com todo o processo em código.

6.4 Decision Tree

Em linhas gerais, o algoritmo da Árvore de Decisão consiste em treinar um modelo que realiza predições baseado em regras lógicas criadas a partir do treino. Dessa forma, a predição do algoritmo se dá ao analisar um ou mais atributos e verificar para qual ramo de escolha ele deve prosseguir com a análise até chegar nos nós da árvore onde encontra-se a classe daquele

caminho percorrido.

Para treinar o modelo, utilizou-se o algoritmo da biblioteca *sklearn* e foi feito o treino da árvore de decisão com as 2 bases: base completa e base reduzida (classe com menor desbalanceamento). A Figura 10 apresenta o resultado dos dois treinos.



Accuracy: 0.9702457956015524

Confusion Matrix
[[8289 117]
[182 1461]]

Classification report

	precision	recall	f1-score	support
0.0	0.978515	0.986081	0.982284	8406
1.0	0.925856	0.889227	0.907172	1643
accuracy			0.970246	10049
macro avg	0.952185	0.937654	0.944728	10049
weighted avg	0.969905	0.970246	0.970003	10049

A

Accuracy: 0.999205675443023

Confusion Matrix
[[840463 119]
[550 1093]]

Classification report

	precision	recall	f1-score	sup
0.0	0.999346	0.999858	0.999602	84
1.0	0.901815	0.665247	0.765674	
accuracy			0.999206	84
macro avg	0.950581	0.832552	0.882638	84
weighted avg	0.999156	0.999206	0.999146	84

B

Figura 10. Árvore de Decisão - Comparação entre base reduzida (A) e base completa (B)

6.5 k-Nearest Neighbors

Em linhas gerais, o algoritmo de k-NN busca separar as classes em clusters e, ao realizar a predição, encontrar o cluster ao que o dado de entrada pertence. Para realizar esta tarefa, o algoritmo executa algum cálculo de distância entre pontos.

Como se trata de um algoritmo com alta variedade de parâmetros, optou-se por deixar os padrões da própria biblioteca e alterar apenas o número de k vizinhos. Para descobrir o melhor k para o dataset, rodou-se um algoritmo simples de iteração entre diferentes k's para comparar o resultado baseado no número de casos verdadeiro positivos (fraudes).

```
Ks = 26
mean_acc = np.zeros((Ks-1))
for n in range(1,Ks):
    neigh = KNeighborsClassifier(n_neighbors=n, n_jobs=3).fit(X_train,y_train)
    y_pred = neigh.predict(X_test)
    mean_acc[n-1] = confusion_matrix(y_test, y_pred)[1][1]
```

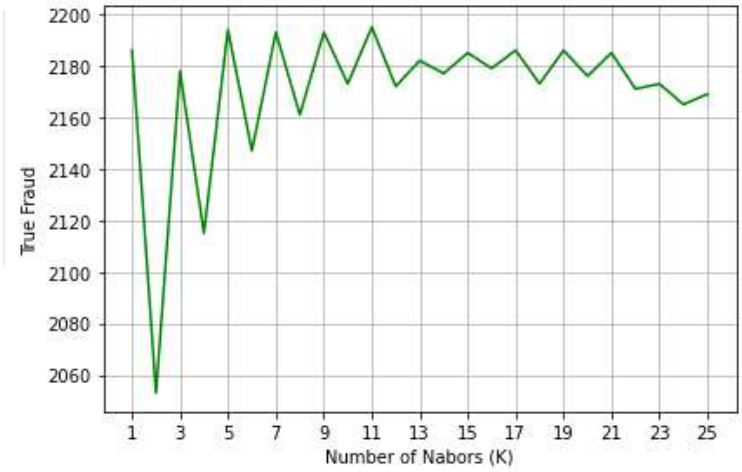


Figura 11. Comparação de True Fraud entre diferentes k's

Ao observar a Figura 11, nota-se que o melhor para k é 5, 7, 9 ou 11. Após este valor, nota-se uma queda na quantidade de acertos. Por esse motivo, o modelo final foi treinado com o valor de k=5 e obteve os resultados demonstrados na Figura 12.

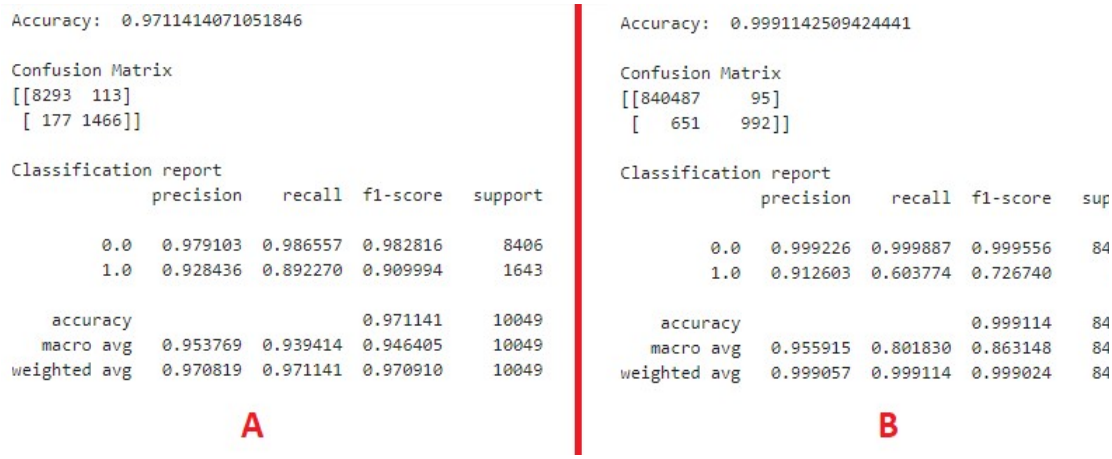


Figura 12. k-NN - Comparação entre base reduzida (A) e base completa (B)

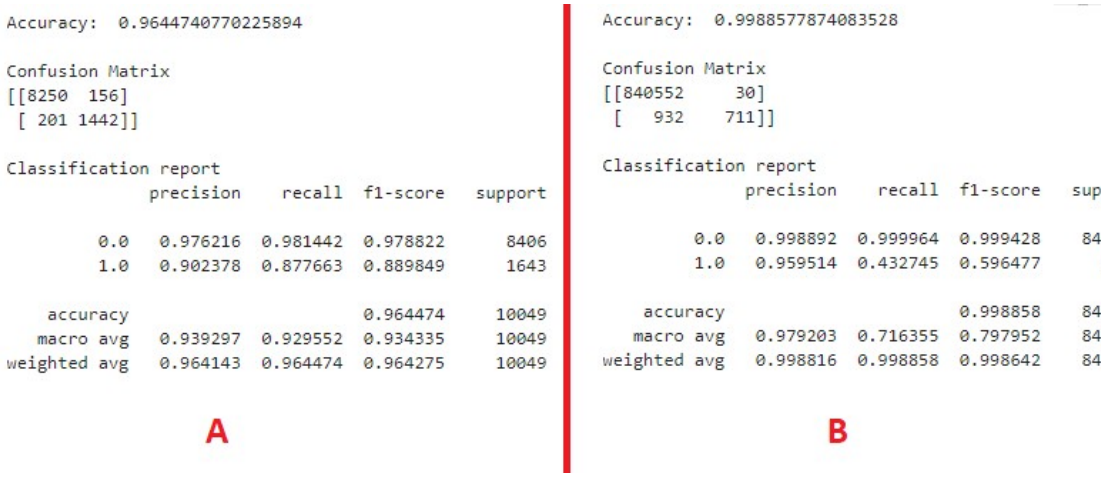
6.6 Multi-Layer Perceptron

O MLP consiste de um algoritmo que aprende a encontrar a classe alvo através da criação de uma função e uma constante alfa. Além disso, como o próprio nome diz, se trata de um classificador multi-camadas. Por fim, existem diferentes métodos para realizar o aprendizado.

Como se trata de um algoritmo complexo e com muitas combinações possíveis de parâmetros, utilizou-se o método de busca de parâmetro ótimo chamado GridSearch [9.4]. Este método testa a combinação entre os possíveis parâmetros e retorna as melhores configurações baseada no critério de aceite escolhido. Novamente, como não estamos olhando para a acurácia, o método de avaliação escolhido foi o *recall*.

Assim, o modelo foi treinado com camadas 3x1000, função de ativação logística, método ADAM, um alfa inicial de 0.0005 e um máximo de 1000 iterações. Porém, deve-se destacar que este parâmetros ótimos foram encontrados baseados nas possibilidades montadas para a execução do GridSearch. E, devido a limitações de hardware, não foi possível testar mais possibilidades.

A Figura 13 demonstra o resultado comparativo entre os modelos com base reduzida e completa.



```
Accuracy:  0.9644740770225894

Confusion Matrix
[[8250  156]
 [ 201 1442]]

Classification report
              precision    recall  f1-score   support

         0.0   0.976216  0.981442  0.978822      8406
         1.0   0.902378  0.877663  0.889849      1643

    accuracy                       0.964474     10049
   macro avg   0.939297  0.929552  0.934335     10049
weighted avg   0.964143  0.964474  0.964275     10049
```

A

```
Accuracy:  0.9988577874083528

Confusion Matrix
[[840552     30]
 [   932    711]]

Classification report
              precision    recall  f1-score   sup

         0.0   0.998892  0.999964  0.999428    84
         1.0   0.959514  0.432745  0.596477

    accuracy                       0.998858    84
   macro avg   0.979203  0.716355  0.797952    84
weighted avg   0.998816  0.998858  0.998642    84
```

B

Figura 13. MLP - Comparação entre base reduzida (A) e base completa (B)

7. Discussão dos Resultados

Nesta seção estão consolidados os resultados apresentados na seção 6. Ao comparar a matriz de confusão entre todos os modelos treinados, nota-se que houve diferença nos resultados obtidos. A Figura 14 demonstra esta comparação em termos das predições relacionadas à classe fraudulenta.

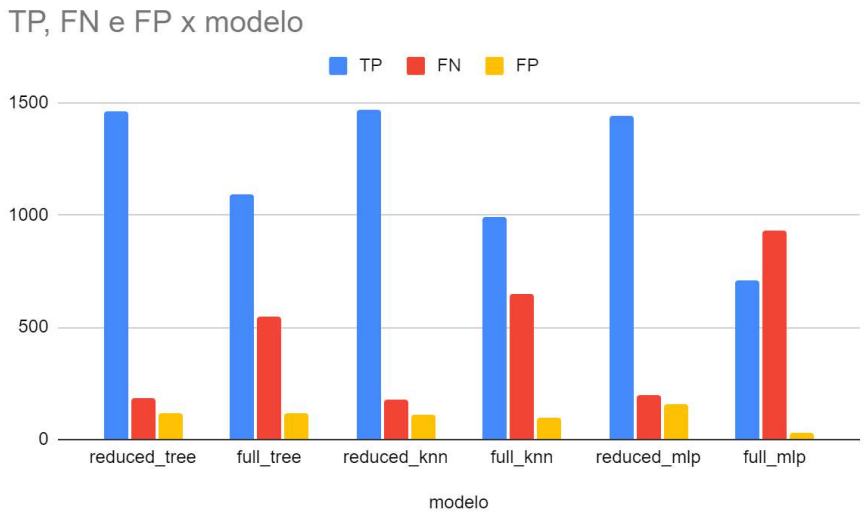


Figura 14. Comparação de TP, FN e FP de cada modelo

O primeiro ponto claro é a queda de predições corretas (TP, em azul) quando troca-se o modelo treinado com o dataset reduzido (*reduced_*) e o modelo treinado com o dataset completo (*full_*). Consequentemente, o número de predições erradas para o pior caso, onde o modelo não encontra fraude quando deveria encontrar (FN, em vermelho), aumentou.

Uma possível explicação para este resultado é o aumento gigantesco de dados deixando a classe fraude com menos de 0.5% dos registros. Nesse caso, realmente é plausível e esperado que o modelo não consiga desenvolver as funções necessárias para esta predição. Inclusive, este resultado reforça a necessidade de lidar com o desbalanceamento de classes.

Já a Figura 15 apresenta a mesma comparação de modelos, porém levando em consideração as métricas de *precision* e *recall*. Obviamente, como o *recall* possui o número de FN em sua fórmula, é esperado que ele tenha uma redução nos casos dos modelos treinados com o dataset completo.

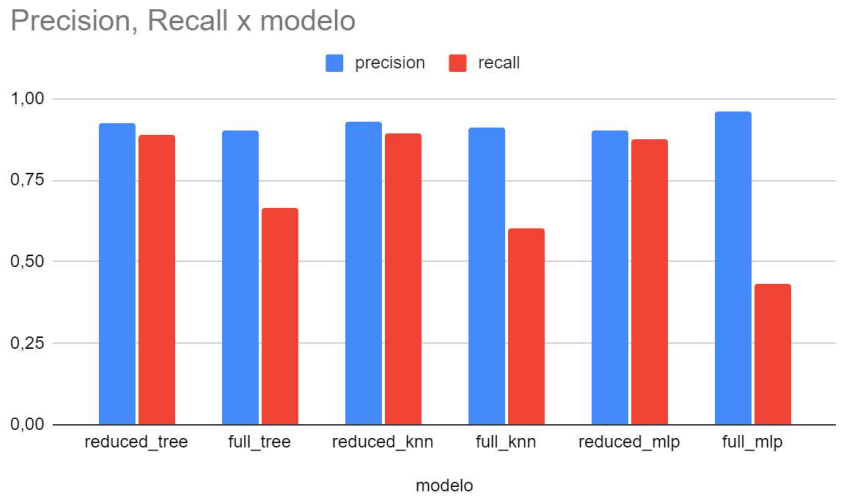


Figura 15. Comparação de Precision e Recall de cada modelo

Ao olhar para as Figuras 14 e 15, é fácil observar que os modelos tiveram um resultado bem semelhante entre si, em relação a classe de fraude. Ou seja, nenhum modelo teve uma disparidade seja no acerto da classe ou no tipo do erro (FN e FP).

8. Conclusão e Trabalhos Futuros

8.1 Conclusão

Pode-se tirar algumas conclusões do resultado obtido neste projeto. O primeiro ponto de destaque está relacionado a análise do dataset. Neste contexto, é possível observar que a qualidade do dataset afetou diretamente a qualidade de predição do modelo: utilizar um dataset extremamente desbalanceado gerou modelos com resultados bem ruins para predizer a classe desbalanceada. Por isso, é sempre importante realizar um bom pré-processamento no dataset antes de definir o algoritmo e os parâmetros utilizados.

Outra consideração está relacionada a escolha do algoritmo para predição. É possível atingir resultados semelhantes em diferentes modelos e não apenas em modelos mais sofisticados e complexos como o MLP. Este tipo de resultado significa que modelos mais simples e com maior explicabilidade podem ser utilizados tão bem quanto outros.

Então, ao analisarmos um dataset desbalanceado em que não podemos aumentar o tamanho da classe desbalanceada e sabemos que o resultado não será satisfatório, é válido realizar uma análise se a qualidade de modelos mais complexos se manterá próxima a qualidade de modelos mais simples. Se este for o caso, a utilização de modelos preditivos mais explicativos auxilia na tomada de decisão, não só evitando uma maior quantidade de erros, como também ajudando na interpretação e entendimento do problema por parte do usuário. É válido refletir sobre o foco na capacidade de explicabilidade do modelo para a tomada de decisão se tivermos resultado semelhante entre os modelos ao aplicar este conceito em produção.

Trabalhar com dataset desbalanceado pode ser um problema ainda sem solução perfeita. Entretanto, se houver brecha para aplicar uma forma de interpretar o resultado do modelo, suas predições falsas poderão ser corrigidas pelo usuário e aplicadas corretamente.

8.2 Limitações

É importante destacar que o objetivo deste trabalho era explorar o problema de dataset desbalanceado para a detecção de fraudes e como a explicabilidade de modelos simples poderia apoiar na tomada de decisão. Por este motivo, o trabalho possui diversas limitações:

- Dataset foi balanceado ao remover registros e nenhum teste foi conduzido com adição de registros;
- GridSearch utilizado para busca de parâmetros ótimos só foi realizado de forma limitada por limitações de *hardware*
- Modelos de redes mais profundas e algoritmos mais complexos que MLP não foram testados até o final também por limitações de

hardware

- Todo o resultado extraído deste projeto está relacionado, não só a estas limitações, como também a limitações do próprio dataset

8.3 Trabalhos Futuros

Devido às limitações apresentadas na seção 8.2, fica claro que o projeto ainda poderia ser muito mais explorado:

- Como seria o resultado se fosse utilizado um método de aumento de registros como, por exemplo, o SMOTE [REF]?
- Outros algoritmos de rede neurais teriam apresentado resultados bem melhores?
- Buscas de hiper-parâmetros mais completas teriam apresentado outros parâmetros e, como consequência, um melhor resultado para o MLP?

Muitos desafios continuam abertos no campo de Machine Learning. Estes desafios se tornam ainda mais claros e problemáticos ao tentar aplicar conceitos com dados presentes no dia a dia. Então, é necessário realizar estudos e testes mais completos.

9. Links

- https://www.kaggle.com/
- https://www.kaggle.com/datasets/vardhansiramdasu/fraudulent-transactions-prediction
- https://python.org
- https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
- Github contendo notebook completo com toda a exploração, análise do dataset, treinamento e avaliação dos modelos: https://github.com/romponciano/fraud-transfer-detection

- Link para base de dados utilizada: https://www.kaggle.com/datasets/vardhansiramdasu/fraudulent-transactions-prediction

10. Referências

[1] Awoyemi, J; Adetunmbi, A.; Oluwadare, S. **Credit card fraud detection using Machine Learning Techniques: A Comparative Analysis**. Nigeria. IEEE, 2017

[2] FBI Report, **Increased Use of Mobile Banking Apps Could Lead to Exploitation.** Estados Unidos, 2020.

[3] Padin, G. **Pix: após alta de crimes, veja como se proteger de golpes e sequestros**. Brasil. R7, 2021

[4] Máximo, W. **Pix terá medidas de segurança para coibir sequestros e roubos**. Brasil. Agência Brasil, 2021

[5] Ganganwar, V. **An overview of classification algorithms for imbalanced datasets**. India, 2012

[6] Varmedja, D *et. al.* **Credit Card Fraud Detection - Machine Learning methods**. Sérvia, 2019

[7] Makki, S. *et. al.* **An Experimental Study With Imbalanced Classification Approaches for Credit Card Fraud Detection**. IEEE, 2019

[8] Perols, J. **Financial Statement Fraud Detection: An Analysis of Statistical and Machine Learning Algorithms**. American Accounting Association, 2011

[9] Gunning, D. *et. al* **XAI—Explainable artificial intelligence**. Science Robotics, 2019

[10] Amina, A.; Berrada, M. **Peeking inside the black-box: a survey on explainable artificial intelligence (XAI).** IEEE, 2018

[11] Wojciech, S; Wiegand, T; Müller, K. **Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models.** arXiv, 2017

[12] Kumar, P. *et. al.* **Credit Card Fraudulent Detection Using Machine**

Learning Algorithm. IJREAM, 2020

[13] Arun, D.; Rad, P. **Opportunities and challenges in explainable artificial intelligence (xai): A survey**. arXiv, 2020

[14] Sebastian, R; Patterson, J; Nolet, C. **Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence.** Information, 2020

[15] Kumar, S. *et. al.* **An Improved Fuzzy Min–Max Neural Network for Data Classification.** IEEE, 2019

[16] Pozzolo, A.; Caelen, O.; Bontempi, G. **When is Undersampling Effective in Unbalanced Classification Tasks?** ECML PKDD, 2015

[17] Pozzolo, A. *et. al.* **Calibrating Probability with Undersampling for Unbalanced Classification.** IEEE, 2015

[18] Krawczyk, B. **Learning from imbalanced data: open challenges and future directions.** Prog Artif Intell, 2016.