Mecanismo de Predição de Fraudes Financeiras Utilizando Aprendizado de Máquina e Processamento Distribuído

Daniel G. M. Lira (daniel.gleison@aluno.uece.br) [1]

Resumo:

O Aprendizado de Máquina ou *Machine Learning* é uma subárea da Inteligência Artificial que fornece às máquinas a capacidade de aprender sem programação explícita e aprimora o tratamento de determinada questão através da experiência, abrangendo os conceitos de ciência da computação, estatística, matemática e ciência de dados. Hoje, a maioria dos sistemas de detecção de fraudes são baseados em conjuntos de regras fixas e previamente definidas. Os sistemas de regras envolvem a criação de critérios "*if-then*" para filtrar as transações, os quais dependem de um conjunto de funções projetadas para identificar tipos específicos de transações de alto risco, utilizando o conhecimento humano para caracterizar operações fraudulentas. Considerando a necessidade de adequação aos novos padrões de ataques, atualmente vem se recorrendo aos modelos de aprendizagem de máquina e processamento distribuído, onde são aplicados algoritmos capazes de realizar a classificação de fraudes com alta acurácia e assertividade, além de menor tempo de resposta, em grandes volumes de dados (*big data*), por meio do técnicas de mineração de dados que envolvem a coleta, limpeza, transformação, balanceamento, treinamento, teste, avaliação e validação das amostras. A proposta desta pesquisa consiste na implementação de mecanismos de predição de fraudes financeiras utilizado os algoritmos de classificação: *Decision Tree, Random Forest, Neural Network Perceptron, Naive Baies, Logistic Regression e Suport Vector Machines*. De acordo com os resultados obtidos, o modelo *Decision Tree* (Árvore de Decisão) apresentou a melhor performance de predição com 99,7% de acurácia, nenhum falso positivo e tempo de resposta de 0,08 segundos[2], tendo como relevante contribuição científica, além da elevada taxa de assertividade, o reduzido tempo de predição para volume massivo de dados.

Palavras-chaves: predição de fraudes, aprendizado de máquina, ciência de dados, processamento distribuído, mineração massiva de dados.

---

[1] Discente do Mestrado Acadêmico em Ciência da Computação da Universidade Estadual do Ceará (UECE).

[2] Link do notebook Python no Github*: http://github.com/danielgleison/data-mining*

1. INTRODUÇÃO

Atualmente, em decorrência do volume significativo das transações eletrônicas gerado pela oferta contínua de produtos e serviços em canais digitais, vimos o rápido crescimento de fraudes nos principais meios de pagamentos, gerando prejuízos financeiros tanto para os clientes como para as instituições bancárias. Ademais, caso a fraude não seja identificada e interrompida tempestivamente, pode-se gerar um efeito cascata com o envolvimento de outros agentes.

Embora existam diversas tecnologias de segurança tais como encriptação dos dados, controle de acesso através de credenciais e pré-cadastramento de dispositivos, ainda existem transações fraudulentas que são concluídas com sucesso nos canais Internet/Mobile Banking. Neste ambiente, existem diversas ameaças que se caracterizam pela complexidade e evolução constante, com vistas a burlar as regras de segurança das empresas. Frequentemente, a evolução tecnológica dos atacantes é mais rápida do que a evolução das empresas, podendo resultar em prejuízos relevantes para as organizações alvejadas [4].

Um sistema de detecção de fraudes eficaz deve ser capaz de detectar transações fraudulentas com alta precisão e eficiência. Porém, além de evitar que atores mal-intencionados executem transações fraudulentas, também é muito importante garantir que usuários autorizados não sejam impedidos de acessar os sistemas de pagamentos [2].

Hoje, a maioria dos sistemas de detecção de fraudes são baseados em conjuntos de regras fixas e previamente definidas. Os sistemas de regras envolvem a criação de critérios “se ... então” para filtrar as transações, os quais dependem de um conjunto de regras projetadas para identificar tipos específicos de transações de alto risco, utilizando o conhecimento humano para caracterizar transações fraudulentas. Normalmente, a eficácia desse tipo de sistema pode ser melhorada com o acréscimo de novas regras ao sistema, entretanto dependerá sobremaneira do conhecimento e experiência do responsável pela definição [6].

Para a automatização desse processo e consequente redução da mão-de-obra e tempo de análise, as grandes indústrias bancárias estão recorrendo aos processos de Inteligência Artificial e Aprendizado de Máquina (Machine Learning), onde são aplicados algoritmos capazes de realizar a classificação de fraudes com alta assertividade e baixo número de falsos positivos, com base na mineração e aprendizagem computacional dos dados históricos.

Diante da necessidade alta capacidade de processamento de máquina para detecção de fraudes em ambientes com Big Data, são utilizados sistemas distribuídos, a exemplo do Apache Spark.

2. FUNDAMENTAÇÃO TEÓRICA

2.1. APRENDIZADO DE MÁQUINA

Aprendizado de Máquina (AM) ou *Machine Learning* (ML) é uma subárea da Inteligência Artificial que fornece às máquinas a capacidade de aprender sem programação explícita e aprimora o tratamento de determinada questão através da experiência. Logo, expande a gama de atividades que podem ser desempenhadas por um computador. Representa uma das áreas técnicas de maior crescimento atualmente, reunindo ciência da computação, estatística, inteligência artificial e ciência de dados [9].

Um algoritmo de aprendizado usa um conjunto de amostras como uma entrada denominada conjunto de treinamento, e se classificados em três categorias principais: aprendizagem supervisionada, não supervisionada e de reforço [10]:

- Aprendizagem supervisionada: desenvolvem o seu modelo de classificação com base em um mapeamento aprendido, capturando os relacionamentos entre os parâmetros de entrada e a saída necessária.
- Aprendizagem não supervisionada: categorizar a entrada dados em grupos distintos, examinando a similaridade entre eles.
- Aprendizagem de reforço: é uma categoria intermediária entre aprendizagem supervisionada e não supervisionada. Os algoritmos são treinados pelos dados de um ambiente com o objetivo de descobrir as melhores abordagens para um determinado agente em diferentes ambientes.

O aprendizado supervisionado ainda se subdivide em três macro áreas [12]:

- Regressão: os algoritmos de AM são utilizados para previsão de comportamento baseado nas variáveis, ou características, usadas como modelo estatístico.
- Classificação: nesta área aplicam-se algoritmos necessários a determinar os grupos aos quais pertencem as amostras presentes no modelo estatístico.
- *Clustering*: os algoritmos aplicados têm o objetivo de determinar quantos e quais são os grupos distintos a que pertencem as amostras do modelo.

Os classificadores podem ser gerados por diversos algoritmos de AM preditivos, por exemplo [4] [5]:

- Árvore de Decisão (*Decision Tree*): recebem como entrada um objeto, descrito por seu conjunto de atributos, e produzem, através de testes baseados no valor dos atributos, uma decisão que corresponde aos possíveis rótulos;
- Floresta Randômica (*Random Forest*): baseado na combinação de preditores de árvores, de tal forma que cada árvore depende dos valores de um vetor aleatório amostrado de forma independente e com a mesma distribuição para todas as árvores na floresta;
- *Naive Bayes*: baseia-se no Teorema de Bayes para cálculo das probabilidades das hipóteses e estima a classificação de novos objetos;
- Regressão Logística (*Logistic Regression*): busca prever a relação entre as variáveis dependentes e independentes, possibilitando assim prever qual será o valor da variável dependente por meio as variáveis independentes disponíveis;
- Redes Neurais Perceptron (*Neural Network Perceptron*): é outro método baseado em otimização de funções que estimam o erro entre as respostas da rede e os rótulos dos objetos, em uma das suas possíveis implementações;
- Máquinas de Vetores de Suporte (*Suport Vector Machines*): esta técnica envolve a solução de um problema de otimização quadrática, formulado com o objetivo de maximizar a margem de separação entre os objetos de diferentes classes.

2.2. PROCESSAMENTO DISTRIBUÍDO

Apache Spark é um mecanismo unificado projetado para processamento de dados distribuídos em grande escala (*big data*), localmente em data centers ou na nuvem.

O Spark incorpora bibliotecas de aprendizado de máquina (MLlib), SQL para consultas interativas (Spark SQL), processamento de stream (*Structured Streaming*) para interação com dados em tempo real e processamento de gráfico (GraphX), permitindo, inclusive, programação em três linguagens: Java, Scala e Python.

A filosofia de design do Spark gira em torno de quatro características principais:

- Velocidade;
- Facilidade de uso;
- Modularidade;
- Extensibilidade.

3. TRABALHOS RELACIONADOS

Os trabalhos [4], [6] , [7] ,[5], [8] e [11], que envolvem detecção de fraudes, ciência de dados e aprendizado de máquina, não citam a utilização de processamento distribuído de dados em grande escala, gerando importante lacuna para a evolução de pesquisas de aperfeiçoamento dos modelos de predição.

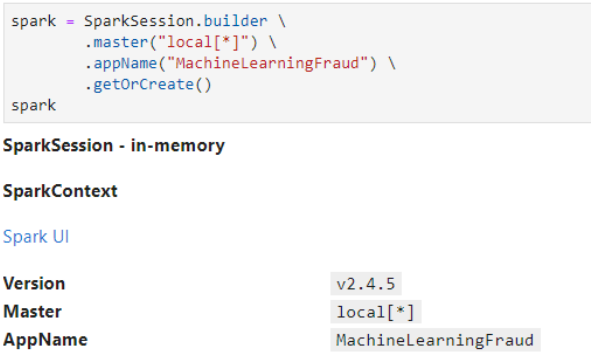
Observa-se, ainda, que o assunto é alvo de interesse em diferentes programas de pós-gradução, tais como: Ciência de Computação no trabalhos [4], [6] e [8], Economia nos trabalhos [5] e [7], além de Engenharia Elétrica em [11] e Mecatrônica Industrial (formação do autor), abrindo-se, portanto, espaço para o fomento de pesquisas conjuntas entre os cientistas de dados e os especialistas do domínio de aplicação.

4. APRESENTAÇÃO DA PROPOSTA

A proposta de pesquisa consiste na implementação de mecanismo de predição de fraudes financeiras utilizado aprendizado de máquina supervisionado de classificação. Especificamente, serão realizados a modelagem, testes e avaliação dos algoritmos: *Decision Tree, Random Forest, Neural Network Perceptron, Naive Baies, Logistic Regression* e *Suport Vector Machines*.

Os modelos foram executados utilizando o framework de computação distribuída Apache Spark versão 2.4.5 (Figura 1), disponível no cluster do LASID/UECE [3](Laboratório de Sistemas Digitais da Universidade Estadual do Ceará). Os scripts foram desenvolvidos em Python no ambiente web do Jupyter Notebook.

Figura 1 - Versão do Apache Spark



Fonte: Elaborada pelo autor

[3] http://lasidhub.uece.br.

Considerando a inviabilidade de utilização de dados financeiros reais, utilizaremos como *dataset* de pesquisa o simulador *PaySim - Synthetic Financial Datasets For Fraud Detection*[4], que dispõe de 6.353.307 transações [1].

O processo de mineração de dados compreende as etapas a seguir (Figura 2):

- Caracterização do problema;
- Coleta dos dados;
- Análise exploratória;
- Preparação dos dados;
- Modelagem;
- Avaliação;
- Apresentação;
- Implantação em produção.

4.1. METODOLOGIA DE AVALIAÇÃO

As métricas utilizadas para avaliação do mecanismo de classificação foram [6]:

- Verdadeiro Positivo (TP): registros positivos previstos corretamente pelo modelo;
- Falso Negativo (FN): registros positivos previstos erroneamente como negativos pelo modelo;
- Falso Positivo (FP): registros negativos previstos erroneamente como positivos pelo modelo;
- Verdadeiro Negativo (TN): registros negativos previstos corretamente pelo modelo;
- Acurácia (*Accuracy*): é a razão entre a quantidade de casos corretamente classificados e todos os casos que passaram pelo classificador [(VP + VN) / (VP + VN + FP + VN)];
- Precisão (*Precision*): determina a fração de registros que realmente são positivos no grupo que o classificador declarou como classe positiva [VP / (VP + FP)];
- Revocação (*Recall*): mede a fração de exemplos positivos previstos

[4] http://www.kaggle.com/ntnu-testimon/paysim1

corretamente pelo classificador [VP / (VP + FN)];

- F1 Score: representa a média harmônica entre precisão e recall [(2* VP) / (2* VP + FP + FN)];
- Matriz de Confusão (MC): tabela que avalia o desempenho de um modelo de classificação baseada na contagem de registros de testes previstos corretamente e incorretamente (Tabela 1).

Tabela 1 - Matriz de Confusão

		Valor Predito	
		0 - Não	1 - Sim
Valor Esperado	0 - Não	Verdadeiro Negativo (VN)	Falso Positivo (FP)
	1 - Sim	Falso Negativo (FN)	Verdadeiro Positivo (VN)

Fonte: Elaborada pelo autor

5. ANÁLISES E RESULTADOS

5.1. COLETA E COMPREENSÃO DOS DADOS

Inicialmente, foi realizada a importação do *dataset* de transações financeiras, em formato CSV, para o dataframe Spark "df_original" (Figura 3).

Figura 2 - Importação do dataset

```
df_original = spark.read.format('csv').options(sep=',',header='true',inferschema='true').\
            load(data_path+'PS_20174392719_1491204439457_log.csv')
df_original.show(5)
```

```
+----+--------+--------+-----------+-------------+--------------+-----------+--------------+--------------+-------+--------------+
|step|    type|  amount|   nameOrig|oldbalanceOrg|newbalanceOrig|   nameDest|oldbalanceDest|newbalanceDest|isFraud|isFlaggedFraud|
+----+--------+--------+-----------+-------------+--------------+-----------+--------------+--------------+-------+--------------+
|   1| PAYMENT| 9839.64|C1231006815|     170136.0|     160296.36|M1979787155|           0.0|           0.0|      0|             0|
|   1| PAYMENT| 1864.28|C1666544295|      21249.0|      19384.72|M2044282225|           0.0|           0.0|      0|             0|
|   1|TRANSFER|   181.0|C1305486145|        181.0|           0.0| C553264065|           0.0|           0.0|      1|             0|
|   1|CASH_OUT|   181.0| C840083671|        181.0|           0.0|  C38997010|       21182.0|           0.0|      1|             0|
|   1| PAYMENT|11668.14|C2048537720|      41554.0|      29885.86|M1230701703|           0.0|           0.0|      0|             0|
+----+--------+--------+-----------+-------------+--------------+-----------+--------------+--------------+-------+--------------+
only showing top 5 rows
```

Fonte: Elaborada pelo autor

O dataset utilizado foi criado em 31/03/2017, a partir de um simulador de dados fictícios, e possui 6.362.620 transações financeiras, das quais 8.213 (0,13% do total) foram registradas como fraudulentas (isFraud = 1).

Na Tabela 3 encontra-se a descrição dos 11 atributos que compõem o dataset, sendo 3 do tipo *integer*, 5 *double* e 3 *string*.

Figura 3 - Tipos de atributos do dataset

```
df_original.printSchema()
```

```
root
 |-- step: integer (nullable = true)
 |-- type: string (nullable = true)
 |-- amount: double (nullable = true)
 |-- nameOrig: string (nullable = true)
 |-- oldbalanceOrg: double (nullable = true)
 |-- newbalanceOrig: double (nullable = true)
 |-- nameDest: string (nullable = true)
 |-- oldbalanceDest: double (nullable = true)
 |-- newbalanceDest: double (nullable = true)
 |-- isFraud: integer (nullable = true)
 |-- isFlaggedFraud: integer (nullable = true)
```

Fonte: Elaborada pelo autor

Tabela 2 - Descrição dos atributos do dataset

Atributo	**Descrição**
step	maps a unit of time in the real world. In this case 1 step is 1 hour of time. Total steps 744 (30 days simulation).
type	CASH-IN, CASH-OUT, DEBIT, PAYMENT and TRANSFER.
amount	amount of the transaction in local currency.
nameOrig	customer who started the transaction
oldbalanceOrg	initial balance before the transaction
newbalanceOrig	new balance after the transaction
nameDest	customer who is the recipient of the transaction
oldbalanceDest	initial balance recipient before the transaction. Note that there is not information for customers that start with M (Merchants).
newbalanceDest	new balance recipient after the transaction. Note that there is not information for customers that start with M (Merchants).
isFraud	This is the transactions made by the fraudulent agents inside the simulation. In this specific dataset the fraudulent behavior of the agents aims to profit by taking control or customers accounts and try to empty the funds by transferring to another account and then cashing out of the system.
isFlaggedFraud	The business model aims to control massive transfers from one account to another and flags illegal attempts. An illegal attempt in this dataset is an attempt to transfer more than 200.000 in a single transaction.

Fonte: [1]

5.2. ANÁLISE EXPLORATÓRIA

Para análise exploratória dos dados, foram geradas as métricas estatísticas de contagem, média, desvio padrão, mínimo e máximo e correlação dos atributos do dataframe

original, conforme Figuras 4, 5 e 6.

Figura 4 - Análise estatística do dataset

```
df_original.describe().toPandas().transpose()
```

	0	1	2	3	4
summary	count	mean	stddev	min	max
step	6362620	243.39724563151657	142.33197104912983	1	743
type	6362620	None	None	CASH_IN	TRANSFER
amount	6362620	179861.90354913066	603858.2314629353	0.0	9.244551664E7
nameOrig	6362620	None	None	C1000000639	C999999784
oldbalanceOrg	6362620	833883.104074487	2888242.673037555	0.0	5.958504037E7
newbalanceOrig	6362620	855113.6685785889	2924048.502954267	0.0	4.958504037E7
nameDest	6362620	None	None	C1000004082	M999999784
oldbalanceDest	6362620	1100701.666519649	3399180.1129944758	0.0	3.5601588935E8
newbalanceDest	6362620	1224996.3982019306	3674128.9421196394	0.0	3.5617927892E8
isFraud	6362620	0.001290820448180152	0.035904796801604175	0	1
isFlaggedFraud	6362620	2.514687345778 94E-6	0.0015857747057365504	0	1

Fonte: Elaborada pelo autor

Figura 5 - Matriz de correlação

```
df_original.toPandas().corr()
```

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
step	1.000000	0.022373	-0.010058	-0.010299	0.027665	0.025888	0.031578	0.003277
amount	0.022373	1.000000	-0.002762	-0.007861	0.294137	0.459304	0.076688	0.012295
oldbalanceOrg	-0.010058	-0.002762	1.000000	0.998803	0.066243	0.042029	0.010154	0.003835
newbalanceOrig	-0.010299	-0.007861	0.998803	1.000000	0.067812	0.041837	-0.008148	0.003776
oldbalanceDest	0.027665	0.294137	0.066243	0.067812	1.000000	0.976569	-0.005885	-0.000513
newbalanceDest	0.025888	0.459304	0.042029	0.041837	0.976569	1.000000	0.000535	-0.000529
isFraud	0.031578	0.076688	0.010154	-0.008148	-0.005885	0.000535	1.000000	0.044109
isFlaggedFraud	0.003277	0.012295	0.003835	0.003776	-0.000513	-0.000529	0.044109	1.000000

Fonte: Elaborada pelo autor

Figura 6 – Comparativo isFraud x isFlaggedFraud

```
df_original.count()
```

```
6362620
```

```
df_original.groupBy('IsFraud').count().show()
```

```
+-------+-------+
|IsFraud|  count|
+-------+-------+
|      1|   8213|
|      0|6354407|
+-------+-------+
```

```
df_original.groupBy('isFlaggedFraud').count().show()
```

```
+--------------+-------+
|isFlaggedFraud|  count|
+--------------+-------+
|             1|     16|
|             0|6362604|
+--------------+-------+
```

Fonte: Elaborada pelo autor

A partir das informações ora analisadas, podemos aferir que:

- O dataset é muito desbalanceado, pois temos 6.354.407 transações sem fraudes e apenas 8.213 com fraude;
- O dataset não é padronizado, pois existe grande variação entre as médias e desvios padrão;
- O dataset não é normalizado, pois os valores não estão no intervalo de 0 a 1;
- O dataset possui 3 campos de texto (type, nameOrig e nameDest) que precisam ser convertidos para formato numérico;
- O dataset não possui valores ausentes ou nulos;
- Os atributos “isFraud” e “isFlaggedFraud” são constituídos de 2 classes (0 e 1), sendo 1 para fraude e 0 para não fraude.
- Não existem atributos que demostrem forte (próxima de 1) ou fraca correlação (próxima de -1) com a classe “isFraud”;
- A maior correlação da classe “isFraud” é com o atributo “amount”, de 0.076;
- Das 8.213 fraudes registradas, o sistema de regra fixa representado pelo atributo “isFlaggedFraud” identificou apenas 16 transações fraudulentas.

5.3. INDEXAÇÃO DOS ATRIBUTOS

As bibliotecas de Machine Learning não aceitam atributos de texto, então faz necessária a indexação do valor textual para um respectivo valor numérico dos atributos “type” e “isFraud”.

No dataframe “df_indexado”, serão criadas 2 novas colunas “indexType” e “label” contendo os valores números indexados das respectivas colunas originais, conforme Figura 7.

Figura 7 - Indexação dos atributos

```
indexer = StringIndexer(inputCol='type', outputCol='indexType').fit(df_selecionado)
df_indexado = indexer.transform(df_original)

indexer = StringIndexer(inputCol='isFraud', outputCol='label').fit(df_selecionado)
df_indexado = indexer.transform(df_indexado)

labelReverse = IndexToString().setInputCol('label')
indexTypeReverse = IndexToString().setInputCol('indexType')

df_indexado.show(5)
```

```
+----+--------+--------+-----------+-------------+--------------+-----------+-------------+-------------+------+-------------+---------+-----+
|step|    type|  amount|   nameOrig|oldbalanceOrg|newbalanceOrig|   nameDest|oldbalanceDest|newbalanceDest|isFraud|isFlaggedFraud|indexType|label|
+----+--------+--------+-----------+-------------+--------------+-----------+-------------+-------------+------+-------------+---------+-----+
|   1| PAYMENT| 9839.64|C1231006815|     170136.0|     160296.36|M1979787155|          0.0|          0.0|     0|            0|      1.0|  0.0|
|   1| PAYMENT| 1864.28|C1666544295|      21249.0|      19384.72|M2044282225|          0.0|          0.0|     0|            0|      1.0|  0.0|
|   1|TRANSFER|   181.0|C1305486145|        181.0|           0.0| C553264065|          0.0|          0.0|     1|            0|      3.0|  1.0|
|   1|CASH_OUT|   181.0| C840083671|        181.0|           0.0|  C38997010|      21182.0|          0.0|     1|            0|      0.0|  1.0|
|   1| PAYMENT|11668.14|C2048537720|      41554.0|      29885.86|M1230701703|          0.0|          0.0|     0|            0|      1.0|  0.0|
+----+--------+--------+-----------+-------------+--------------+-----------+-------------+-------------+------+-------------+---------+-----+
only showing top 5 rows
```

Fonte: Elaborada pelo autor

5.4. SELEÇÃO DOS ATRIBUTOS

Considerando que as informações dos atributos “nameOrig” e “nameDest” não são relevantes para detecção de fraude, serão excluídos do dataset. O campo “isFlaggedFraud” pode ser excluído sem prejuízo para o modelo, pois a quantidade de fraudes sinalizadas representa apenas 0,2% do total de fraudes registradas. Os atributos “type” e “isFraud” que foram indexados para novas colunas “indexType” e “label” devem ser excluídos do dataset (Figura 8).

Figura 8 - Exclusão de atributos do dataset

```
df_selecionado = df_indexado.drop('type','nameOrig','nameDest','isFraud','isFlaggedFraud')
df_selecionado.show(5)
```

```
+----+--------+-------------+--------------+--------------+--------------+---------+-----+
|step|  amount|oldbalanceOrg|newbalanceOrig|oldbalanceDest|newbalanceDest|indexType|label|
+----+--------+-------------+--------------+--------------+--------------+---------+-----+
|   1| 9839.64|     170136.0|     160296.36|           0.0|           0.0|      1.0|  0.0|
|   1| 1864.28|      21249.0|      19384.72|           0.0|           0.0|      1.0|  0.0|
|   1|   181.0|        181.0|           0.0|           0.0|           0.0|      3.0|  1.0|
|   1|   181.0|        181.0|           0.0|       21182.0|           0.0|      0.0|  1.0|
|   1|11668.14|      41554.0|      29885.86|           0.0|           0.0|      1.0|  0.0|
+----+--------+-------------+--------------+--------------+--------------+---------+-----+
only showing top 5 rows
```

Fonte: Elaborada pelo autor

No dataframe “df_selecionado” foram descartados os atributos irrelevantes para a eficiência dos modelos, restando aqueles que serão efetivamente empregados no aprendizado de máquina, quais sejam: “step”, “amount”, “oldbalanceOrg”, “newbalanceOrig”, “oldbalanceDest”, “newbalanceDest”, “indexType” e “label” (Figura 9).

Figura 9 - Atributos selecionados

```
df_selecionado.show(5)
```

```
+----+--------+-------------+--------------+--------------+--------------+---------+-----+
|step|  amount|oldbalanceOrg|newbalanceOrig|oldbalanceDest|newbalanceDest|indexType|label|
+----+--------+-------------+--------------+--------------+--------------+---------+-----+
|   1| 9839.64|     170136.0|     160296.36|           0.0|           0.0|      1.0|  0.0|
|   1| 1864.28|      21249.0|      19384.72|           0.0|           0.0|      1.0|  0.0|
|   1|   181.0|        181.0|           0.0|           0.0|           0.0|      3.0|  1.0|
|   1|   181.0|        181.0|           0.0|       21182.0|           0.0|      0.0|  1.0|
|   1|11668.14|      41554.0|      29885.86|           0.0|           0.0|      1.0|  0.0|
+----+--------+-------------+--------------+--------------+--------------+---------+-----+
only showing top 5 rows
```

Fonte: Elaborada pelo autor

5.5. TRANSFORMAÇÃO DO DATASET

As bibliotecas de Machine Learning requerem a entrada de 2 colunas: feature

(entrada) e label (saída). A coluna “feature” deve ser composta por um vetor dos atributos de entrada do dataframe “df_selecionado”, enquanto a coluna “label”, o atributo de saída do mesmo dataframe.

A transformação dos atributos de entrada em um vetor é realizada pela função “VectorAssembler” do Spark. A matriz de classificação “df_trasformado” (6362620 x 2) encontra-se na Figura 10.

Figura 10 - Matriz de classificação

```
ignore = ['label']
list = [x for x in df_selecionado.columns if x not in ignore]

assembler = VectorAssembler(
            inputCols= list,
            outputCol='features')

df_transformado = (assembler.transform(df_selecionado).select('label','features'))
df_transformado.show(truncate = False, n = 5)
```

```
+-----+-----------------------------------------+
|label|features                                 |
+-----+-----------------------------------------+
|0.0  |[1.0,9839.64,170136.0,160296.36,0.0,0.0,1.0]|
|0.0  |[1.0,1864.28,21249.0,19384.72,0.0,0.0,1.0]  |
|1.0  |[1.0,181.0,181.0,0.0,0.0,0.0,3.0]           |
|1.0  |[1.0,181.0,181.0,0.0,21182.0,0.0,0.0]       |
|0.0  |[1.0,11668.14,41554.0,29885.86,0.0,0.0,1.0] |
+-----+-----------------------------------------+
only showing top 5 rows
```

Fonte: Elaborada pelo autor

5.6. DIVISÃO DO DATASET

A matriz de classificação foi dividida em 2 partas de forma aleatória, sendo 70% dos dados destinados ao dataframe de treinamento (“train”), totalizando 4.453.834 registros e 30% para o dataframe de teste (“test”), totalizando 1.908.786 registros (Figura 11). Para tanto, utilizou-se a função “randomSplit” do Spark.

Figura 11 - Divisão do dataset

```
train_sample = 0.7
test_sample = 0.3

(train, test) = df_transformado.randomSplit([train_sample, test_sample],1234)

num_train = df_transformado.count() * train_sample
num_test = df_transformado.count() * test_sample

print('Percentual da base de treinamento', train_sample*100, '%')
print('Percentual da base de teste', test_sample*100, '%')
print('Quantidade de registros da base de treinamento', num_train)
print('Quantidade de registros da base de treinamento', num_test)
```

```
Percentual da base de treinamento 70.0 %
Percentual da base de teste 30.0 %
Quantidade de registros da base de treinamento 4453834.0
Quantidade de registros da base de treinamento 1908786.0
```

Fonte: Elaborada pelo autor

A quantidade de fraudes (label = 1.0) no dataset de teste foi de 2.482, representando 0,0013% do total (Figura 12).

Figura 12 - Quantidade de fraudes do dataset de teste

```
test.groupby('label').count().show()
```

```
+-----+-------+
|label|  count|
+-----+-------+
|  0.0|1903395|
|  1.0|   2482|
+-----+-------+
```

Fonte: Elaborada pelo autor

5.7. AVALIAÇÃO DOS ALGORITMOS

Neste trabalho utilizaremos os algoritmos de classificação: Decision Tree, Radom Forest, Neural Network Perceptron, Naive Bayes, Logistic Regression e Suport Vector Machines (SVM).

Para avaliação dos modelos, foram analisadas as seguintes variáveis: acurácia, recall, precisão, F1, quantidade de falsos positivos, quantidade de falsos negativos, tempo de treinamento e tempo de predição.

5.7.1. DECISION TREE (DT)

No algoritmo Decision Tree foi utilizado o classificador “DecisionTreeClassifier” da biblioteca MLlib do Spark. A acurácia do modelo foi de 99,93%, enquanto a quantidade de falsos positivos (FP) e falsos negativos (FN) foram de 201 e 1.117, respectivamente. O tempo total de execução (treinamento e predição) foi de 18,63 segundos.

Na Figura 13 e Tabela 4 encontram-se representadas a matriz de confusão e na Figura 14 a avaliação completa do algoritmo DT.

Figura 13 - Matriz de confusão do modelo DT

```
# Matriz de confusão
y_true = result_dt.select("label").toPandas()
y_pred = result_dt.select("prediction").toPandas()
mc_dt = confusion_matrix(y_true, y_pred)
tn_dt, fp_dt, fn_dt, tp_dt = confusion_matrix(y_true, y_pred).ravel()
print(mc_dt)
```

```
[[1903194     201]
 [   1177    1305]]
```

Fonte: Elaborada pelo autor

Tabela 3 - Matriz de confusão do modelo DT

	Valor Previsto	
Valor Esperado	0 - Não Fraude	1 - Fraude
0 - Não Fraude	1.903.194 (VN)	201 (FP)
1 - Fraude	1.117 (FN)	1.305 (VP)

Fonte: Elaborada pelo autor

Figura 14 – Avaliação do modelo DT

```
# Exibição dos resultados
evaluator_dt = spark.createDataFrame(
    [(round(accuracy_dt,2), round(recall_dt,2), round(precision_dt,2), round(f1_dt,2),\
      int(fp_dt), int(fn_dt),\
      round(time_dt_train,2), round(time_dt_pred,2))],\
    ['acurácia','recall','precisão','f1 score',\
     'falso positivo', 'falso negativo',\
     'tempo treinamento','tempo predição'])
print("Resultados do modelo Decision Tree:")
evaluator_dt.show()
```

```
Resultados do modelo Decision Tree:
+--------+------+--------+--------+--------------+--------------+-----------------+--------------+
|acurácia|recall|precisão|f1 score|falso positivo|falso negativo|tempo treinamento|tempo predição|
+--------+------+--------+--------+--------------+--------------+-----------------+--------------+
|   99.93| 99.93|   99.92|   99.92|           201|          1177|            18.56|          0.07|
+--------+------+--------+--------+--------------+--------------+-----------------+--------------+
```

Fonte: Elaborada pelo autor

5.7.2. RANDOM FOREST (RF)

No algoritmo Random Forest foi utilizado o classificador "RandomForestClassifier" da biblioteca MLlib do Spark. A acurácia do modelo foi de 99,93%, enquanto a quantidade de falsos positivos (FP) e falsos negativos (FN) foram de 0 e 1.328, respectivamente. O tempo total de execução (treinamento e predição) foi de 45 segundos.

Na Figura 15 e Tabela 5 encontram-se representadas a matriz de confusão e na Figura 16 a avaliação completa do algoritmo RT.

Figura 15 - Matriz de confusão do modelo RF

```
# Matriz de confusão
y_true = result_rf.select("label").toPandas()
y_pred = result_rf.select("prediction").toPandas()
mc_rf = confusion_matrix(y_true, y_pred)
tn_rf, fp_rf, fn_rf, tp_rf = confusion_matrix(y_true, y_pred).ravel()
print(mc_rf)
```

```
[[1903395       0]
 [   1328    1154]]
```

Fonte: Elaborada pelo autor

Tabela 4 - Matriz de confusão do modelo RF

	Valor Previsto	
Valor Esperado	0 - Não Fraude	1 - Fraude
0 - Não Fraude	1.903.395 (VN)	0 (FP)
1 - Fraude	1.328 (FN)	1.154 (VP)

Fonte: Elaborada pelo autor

Figura 16 – Avaliação do modelo RF

```
# Exibição dos resultados
evaluator_rf = spark.createDataFrame(
    [(round(accuracy_rf,2), round(recall_rf,2), round(precision_rf,2), round(f1_rf,2),\
      int(fp_rf), int(fn_rf),\
      round(time_rf_train,2), round(time_rf_pred,2))],\
    ['acurácia','recall','precisão','f1 score',\
     'falso positivo', 'falso negativo',\
     'tempo treinamento','tempo predição'])
print("Resultados do modelo Decision Tree:")
evaluator_rf.show()
```

```
Resultados do modelo Decision Tree:
+--------+------+--------+--------+--------------+--------------+-----------------+--------------+
|acurácia|recall|precisão|f1 score|falso positivo|falso negativo|tempo treinamento|tempo predição|
+--------+------+--------+--------+--------------+--------------+-----------------+--------------+
|   99.93| 99.93|   99.93|   99.92|             0|          1328|            44.91|          0.09|
+--------+------+--------+--------+--------------+--------------+-----------------+--------------+
```

Fonte: Elaborada pelo autor

5.7.3. NEURAL NETWORK PERCEPTRON (NNP)

No algoritmo Neural Network Perceptron foi utilizado o classificador "MultilayerPerceptronClassifier" da biblioteca MLlib do Spark. A acurácia do modelo foi de 99,87%, enquanto a quantidade de falsos positivos (FP) e falsos negativos (FN) foram de 0 e 2.482, respectivamente. O tempo total de execução (treinamento e predição) foi de 34,73 segundos.

Na Figura 17 e Tabela 6 encontram-se representadas a matriz de confusão e na Figura 18 a avaliação completa do algoritmo NNP.

Figura 17 – Matriz de confusão do modelo NNP

```
# Matriz de confusão
y_true = result_nnp.select("label").toPandas()
y_pred = result_nnp.select("prediction").toPandas()
mc_nnp = confusion_matrix(y_true, y_pred)
tn_nnp, fp_nnp, fn_nnp, tp_nnp = confusion_matrix(y_true, y_pred).ravel()
print(mc_nnp)
```

```
[[1903395       0]
 [   2482       0]]
```

Fonte: Elaborada pelo autor

Tabela 5 - Matriz de confusão do modelo NNP

	Valor Previsto	
Valor Esperado	0 - Não Fraude	1 - Fraude
0 - Não Fraude	1.903.395 (VN)	0 (FP)
1 - Fraude	2.482 (FN)	0 (VP)

Fonte: Elaborada pelo autor

Figura 18 - Avaliação do modelo NNP

```
# Exibição dos resultados
evaluator_nnp = spark.createDataFrame(
    [(round(accuracy_nnp,2), round(recall_nnp,2), round(precision_nnp,2), round(f1_nnp,2),\
      int(fp_nnp), int(fn_nnp),\
      round(time_nnp_train,2), round(time_nnp_pred,2))],\
    ['acurácia','recall','precisão','f1 score',\
     'falso positivo', 'falso negativo',\
     'tempo treinamento','tempo predição'])
print("Resultados do modelo Decision Tree:")
evaluator_nnp.show()
```

```
Resultados do modelo Decision Tree:
+--------+------+--------+--------+--------------+--------------+-----------------+--------------+
|acurácia|recall|precisão|f1 score|falso positivo|falso negativo|tempo treinamento|tempo predição|
+--------+------+--------+--------+--------------+--------------+-----------------+--------------+
|   99.87| 99.87|   99.74|    99.8|             0|          2482|            34.67|          0.06|
+--------+------+--------+--------+--------------+--------------+-----------------+--------------+
```

Fonte: Elaborada pelo autor

5.7.4. NAIVE BAYES (NB)

No algoritmo Naive Bayes foi utilizado o classificador “NaiveBayes” da biblioteca MLlib do Spark. A acurácia do modelo foi de 62,03%, enquanto a quantidade de falsos positivos (FP) e falsos negativos (FN) foram de 723.305 e 352, respectivamente. O tempo total de execução (treinamento e predição) foi de 17,03 segundos.

Na Figura 19 e Tabela 7 encontram-se representadas a matriz de confusão e na Figura 20 a avaliação completa do algoritmo NB.

Figura 19 – Matriz de confusão do modelo BN

```
# Matriz de confusão
y_true = result_nb.select("label").toPandas()
y_pred = result_nb.select("prediction").toPandas()
mc_nb = confusion_matrix(y_true, y_pred)
tn_nb, fp_nb, fn_nb, tp_nb = confusion_matrix(y_true, y_pred).ravel()
print(mc_nb)
```

```
[[1180090  723305]
 [    352    2130]]
```

Fonte: Elaborada pelo autor

Tabela 6 - Matriz de confusão do modelo NB

	Valor Previsto	
Valor Esperado	0 - Não Fraude	1 - Fraude
0 - Não Fraude	1.180.090 (VN)	723.305 (FP)
1 - Fraude	352 (FN)	1.305 (VP)

Fonte: Elaborada pelo autor

Figura 20 - Avaliação do modelo NB

```
# Exibição dos resultados
evaluator_nb = spark.createDataFrame(
    [(round(accuracy_nb,2), round(recall_nb,2), round(precision_nb,2), round(f1_nb,2),\
      int(fp_nb), int(fn_nb),\
      round(time_nb_train,2), round(time_nb_pred,2))],\
    ['acurácia','recall','precisão','f1 score',\
     'falso positivo', 'falso negativo',\
     'tempo treinamento','tempo predição'])
print("Resultados do modelo Decision Tree:")
evaluator_nb.show()
```

```
Resultados do modelo Decision Tree:
+--------+------+--------+--------+--------------+--------------+-----------------+--------------+
|acurácia|recall|precisão|f1 score|falso positivo|falso negativo|tempo treinamento|tempo predição|
+--------+------+--------+--------+--------------+--------------+-----------------+--------------+
|   62.03| 62.03|   99.84|   76.43|        723305|           352|            16.97|          0.06|
+--------+------+--------+--------+--------------+--------------+-----------------+--------------+
```

Fonte: Elaborada pelo autor

5.7.5. LOGISTIC REGRESSION (LR)

No algoritmo Logistic Regression foi utilizado o classificador "LogisticRegression" da biblioteca MLlib do Spark. A acurácia do modelo foi de 99,89%, enquanto a quantidade de falsos positivos (FP) e falsos negativos (FN) foram de 15 e 2.165, respectivamente. O tempo total de execução (treinamento e predição) foi de 12,50 segundos.

Na Figura 21 e Tabela 8 encontram-se representadas a matriz de confusão e na Figura 22 a avaliação completa do algoritmo LR.

Figura 21 – Matriz de confusão do modelo LR

```
# Matriz de confusão
y_true = result_lr.select("label").toPandas()
y_pred = result_lr.select("prediction").toPandas()
mc_lr = confusion_matrix(y_true, y_pred)
tn_lr, fp_lr, fn_lr, tp_lr = confusion_matrix(y_true, y_pred).ravel()
print(mc_lr)
```

```
[[1903380      15]
 [   2165     317]]
```

Fonte: Elaborada pelo autor

Tabela 7 - Matriz de confusão do modelo LR

	Valor Previsto	
Valor Esperado	0 - Não Fraude	1 - Fraude
0 - Não Fraude	1.903.380 (VN)	15 (FP)
1 - Fraude	2.165 (FN)	317 (VP)

Fonte: Elaborada pelo autor

Figura 22 – Avaliação do modelo LR

```
# Exibição dos resultados
evaluator_lr = spark.createDataFrame(
    [(round(accuracy_lr,2), round(recall_lr,2), round(precision_lr,2), round(f1_lr,2),\
      int(fp_lr), int(fn_lr),\
      round(time_lr_train,2), round(time_lr_pred,2))],\
    ['acurácia','recall','precisão','f1 score',\
     'falso positivo', 'falso negativo',\
     'tempo treinamento','tempo predição'])
print("Resultados do modelo Decision Tree:")
evaluator_lr.show()
```

```
Resultados do modelo Decision Tree:
+--------+------+--------+--------+--------------+--------------+-----------------+--------------+
|acurácia|recall|precisão|f1 score|falso positivo|falso negativo|tempo treinamento|tempo predição|
+--------+------+--------+--------+--------------+--------------+-----------------+--------------+
|   99.89| 99.89|   99.88|   99.84|            15|          2165|            12.48|          0.02|
+--------+------+--------+--------+--------------+--------------+-----------------+--------------+
```

Fonte: Elaborada pelo autor

5.7.6. SUPORT VECTOR MACHINES (SVM)

No algoritmo Suport Vector Machines foi utilizado o classificador “LinearSVC” da biblioteca MLlib do Spark. A acurácia do modelo foi de 99,87%, enquanto a quantidade de falsos positivos (FP) e falsos negativos (FN) foram de 0 e 2.482, respectivamente. O tempo total de execução (treinamento e predição) foi de 907,54 segundos.

Na Figura 23 e Tabela 9 encontram-se representadas a matriz de confusão e na Figura 24 a avaliação completa do algoritmo SVM.

Figura 23 – Matriz de confusão do modelo SVM

```
# Matriz de confusão
y_true = result_svm.select("label").toPandas()
y_pred = result_svm.select("prediction").toPandas()
mc_svm = confusion_matrix(y_true, y_pred)
tn_svm, fp_svm, fn_svm, tp_svm = confusion_matrix(y_true, y_pred).ravel()
print(mc_svm)
```

```
[[1903395       0]
 [   2482       0]]
```

Fonte: Elaborada pelo autor

Tabela 8 - Matriz de confusão do modelo SVM

	Valor Previsto	
Valor Esperado	0 - Não Fraude	1 - Fraude
0 - Não Fraude	1.903.395 (VN)	0 (FP)
1 - Fraude	2.482 (FN)	0 (VP)

Fonte: Elaborada pelo autor

Figura 24 - Avaliação do modelo SVM

```
# Exibição dos resultados
evaluator_svm = spark.createDataFrame(
    [(round(accuracy_svm,2), round(recall_svm,2), round(precision_svm,2), round(f1_svm,2),\
      int(fp_svm), int(fn_svm),\
      round(time_svm_train,2), round(time_svm_pred,2))],\
    ['acurácia','recall','precisão','f1 score',\
     'falso positivo', 'falso negativo',\
     'tempo treinamento','tempo predição'])
print("Resultados do modelo Decision Tree:")
evaluator_svm.show()
```

```
Resultados do modelo Decision Tree:
+--------+------+--------+--------+--------------+--------------+-----------------+--------------+
|acurácia|recall|precisão|f1 score|falso positivo|falso negativo|tempo treinamento|tempo predição|
+--------+------+--------+--------+--------------+--------------+-----------------+--------------+
|   99.87| 99.87|   99.74|    99.8|             0|          2482|           907.46|          0.08|
+--------+------+--------+--------+--------------+--------------+-----------------+--------------+
```

Fonte: Elaborada pelo autor

5.8. RESULTADOS

Após a execução dos scripts de avaliação, obtivemos os seguintes resultados:

- Maior acurácia: RF com 99,93% (Figura 25);

Figura 25 - Ranking de Acurácia em %

```
+-------------------------+------------------+
|Modelo                   |Acuracia          |
+-------------------------+------------------+
|Random Forest            |99.93032079195038 |
|Decision Tree            |99.92769732779188 |
|Logistic Regression      |99.88561696268961 |
|Neural Network Perceptron|99.86977123917231 |
|Naive Bayes              |62.030235949119486|
|Suport Vector Machines   |62.030235949119486|
+-------------------------+------------------+
```

Fonte: Elaborada pelo autor

- Maior recall: RF com 99,93% (Figura 26);

Figura 26 - Ranking de Recall em %

```
+-------------------------+-----------------+
|Modelo                   |Recall           |
+-------------------------+-----------------+
|Random Forest            |99.93032079195038|
|Decision Tree            |99.92769732779189|
|Logistic Regression      |99.88561696268961|
|Neural Network Perceptron|99.86977123917231|
|Suport Vector Machines   |99.86977123917231|
|Naive Bayes              |62.03023594911949|
+-------------------------+-----------------+
```

Fonte: Elaborada pelo autor

- Maior precisão: RF com 99,93% (Figura 27);

Figura 27 - Ranking de Precisão em %

Modelo	Precisao
Random Forest	99.93036937328648
Decision Tree	99.92089417472314
Logistic Regression	99.88064834117112
Naive Bayes	99.84037310736376
Neural Network Perceptron	99.73971207364609
Suport Vector Machines	99.73971207364609

Fonte: Elaborada pelo autor

- Maior F1 score: DT com 99,92% (Figura 28);

Figura 28 - Ranking de F1 score em %

Modelo	F1
Decision Tree	99.91885912562523
Random Forest	99.91760824566039
Logistic Regression	99.8419528111597
Neural Network Perceptron	99.80469928520955
Suport Vector Machines	99.80469928520955
Naive Bayes	76.43496948710403

Fonte: Elaborada pelo autor

- Menor tempo de treinamento: LT com 12,48 s (Figura 29);

Figura 29 - Ranking de Tempo de Treinamento em segundos

Modelo	Tempo_Treinamento
Logistic Regression	12.480078935623169
Naive Bayes	16.965655088424683
Decision Tree	17.992743015289307
Neural Network Perceptron	34.66543126106262
Random Forest	44.90826463699341
Suport Vector Machines	907.4550714492798

Fonte: Elaborada pelo autor

- Menor tempo de predição: NNP com 0,62 segundos (Figura 30);

Figura 30 - Ranking de Tempo de Predição

Modelo	Tempo_Predicao
Neural Network Perceptron	0.06189155578613281
Naive Bayes	0.06189155578613281
Logistic Regression	0.06189155578613281
Suport Vector Machines	0.06189155578613281
Decision Tree	0.08043575286865234
Random Forest	0.0889892578125

Fonte: Elaborada pelo autor

- Menor quantidade de falsos positivos: NNP, RF e SVM com 0 falsos positivos (Figura 31);

Figura 31 - Ranking da Quantidade de Falsos Positivos

Modelo	Falso_Positivo
Neural Network Perceptron	0
Random Forest	0
Suport Vector Machines	0
Logistic Regression	15
Decision Tree	201
Naive Bayes	723305

Fonte: Elaborada pelo autor

- Menor quantidade de falsos negativos: NB com 352 falsos negativos (Figura 32).

Figura 32 - Ranking da Quantidade de Falsos Negativos

Modelo	Falso_Negativo
Naive Bayes	352
Decision Tree	1177
Random Forest	1328
Logistic Regression	2165
Neural Network Perceptron	2482
Suport Vector Machines	2482

Fonte: Elaborada pelo autor

Na Figura 33, tem-se o quadro comparativo das métricas dos modelos executados de aprendizado de máquina.

Figura 33 – Resultados Geral com dataset original

Modelo	Acuracia	Recall	Precisao	F1	Falso_Positivo	Falso_Negativo	Tempo_Treinamento	Tempo_Predicao
Decision Tree	99.93	99.93	99.92	99.92	201	1177	17.71	0.09
Random Forest	99.93	99.93	99.93	99.92	0	1328	44.91	0.09
Logistic Regression	99.89	99.89	99.88	99.84	15	2165	12.48	0.06
Neural Network Perceptron	99.87	99.87	99.74	99.8	0	2482	34.67	0.06
Suport Vector Machines	62.03	99.87	99.74	99.8	0	2482	907.46	0.06
Naive Bayes	62.03	62.03	99.84	76.43	723305	352	16.97	0.06

Fonte: Elaborada pelo autor

Em razão do desbalanceamento acentuado do dataset, além do nível de acurácia, devemos observar a quantidade de falsos positivos e falsos negativos.

O algoritmo que apresentou a melhor performance foi o *Decison Tree* com acurácia de 99,93%, tempo de treinamento de 17,71s, tempo de predição de 0,09s, 201 falsos positivos (0,01%) e 1.177 falsos negativos (0,06%).

Destaca-se o reduzido tempo de predição (0,09s) proporcionado pela aplicação do processamento distribuído.

5.9. BALANCEAMENTO DO DATASET

Quando alimentados com dados desbalanceados, os algoritmos de ML tendem a gerar modelos que favorecem a classificação de novos dados na classe majoritária (fraud = 0), resultado em excesso de erros na classe minoritária (fraud = 1). Para contornar essa situação, utilizamos a técnica de redefinição do tamanho do conjunto de dados.

Nesse caso, pode-se acrescentar objetos à classe minoritária ou eliminar objetos da classe majoritária. Com o acréscimo de novos objetos, aumenta-se o esforço computacional para cálculo do modelo e há o risco de indução de um modelo inadequado para os dados; além disso, pode ocorrer o superajustamento (overfitting) do modelo aos dados de treinamento. Quando dados são eliminados da classe majoritária, é possível que dados de grande importância para a indução do modelo não sejam considerados, levando ao subajustamento do modelo (underfitting) (RAMOS, 2014).

Conforme Figura 34, foi gerado novo dataset com redução da classe majoritária e balanceamento das classes (50% de fraudes e 50% de não fraudes), totalizando 16.426 registros (0,26% do dataset original).

Figura 34 - Balanceamento do dataset

```
df0 = df_original[df_original.isFraud==0]
df1 = df_original[df_original.isFraud==1]
df0.groupby('isFraud').count().show()
df1.groupby('isFraud').count().show()
```

```
+-------+-------+
|isFraud|  count|
+-------+-------+
|      0|6354407|
+-------+-------+

+-------+-----+
|isFraud|count|
+-------+-----+
|      1| 8213|
+-------+-----+
```

```
guessedFraction = 1.0
noOfSamples = df1.count()
df0 = df0.sample(True, guessedFraction).limit(noOfSamples)
df0.groupby('isFraud').count().show()
```

```
+-------+-----+
|isFraud|count|
+-------+-----+
|      0| 8113|
+-------+-----+
```

```
df_balanceado = df0.union(df1)
df_balanceado.count()
```

```
16426
```

Fonte: Elaborada pelo autor

5.10. VALIDAÇÃO

Foi realizado novo treinamento e teste com o dataset balanceado em todos os algoritmos envolvidos, cujos resultados estão discriminados na Figura 35.

Figura 35 - Resultado Geral com dataset balanceado

```
df = df_acuracia.join(df_fp, "Modelo")
df = df.join(df_fn, "Modelo")
df = df.join(df_time_train, "Modelo")
df = df.join(df_time_pred, "Modelo")
df.sort(df.Acuracia.desc()).show(truncate = False)
```

```
+------------------------+-----------------+--------------+--------------+------------------+-------------------+
|Modelo                  |Acuracia         |Falso_Positivo|Falso_Negativo|Tempo_Treinamento |Tempo_Predicao     |
+------------------------+-----------------+--------------+--------------+------------------+-------------------+
|Decision Tree           |99.69180193137457|0             |15            |9.192930221557617 |0.08043622970581055|
|Random Forest           |99.65070885555784|0             |17            |10.355937004089355|0.10537910461425781|
|Logistic Regression     |98.62338196013971|3             |64            |9.162161350250244 |0.07389688491821289|
|Neural Network Perceptron|90.4253133347031 |410           |56            |58.17394018173218 |0.07389688491821289|
|Naive Bayes             |75.69344565440723|882           |301           |6.018979549407959 |0.07389688491821289|
|Suport Vector Machines  |75.69344565440723|226           |117           |619.2739949226379 |0.07389688491821289|
+------------------------+-----------------+--------------+--------------+------------------+-------------------+
```

Fonte: Elaborada pelo autor

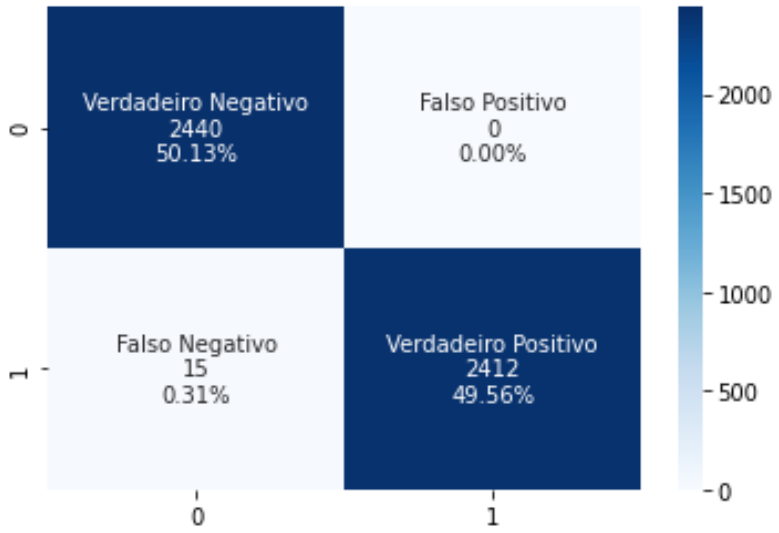
Com o dataset balanceado, o modelo Decision Tree manteve a melhor performance, com acurácia de 99,69%, nenhum falso positivo, 15 falsos negativos, tempo treinamento de 9,19s e treinamento de predição de 0,08s , conforme conforme Tabela 9 e Figura 36, mostrando-se, portanto, válido para utilização em bases de dados desconhecidas.

Tabela 9 - Dataset desbalanceado x Dataset balanceado

	Dataset Desbalanceado	**Dataset Balanceado**
Dataset Total	6.362.620	16.426
Dataset de Treinamento (70%)	4.453.834	11.559
Dataset de Teste (30%)	1.908.786	4.867
Quantidade de Não Fraudes (Classe = 0)	6.354.407	8.213
Quantidade de Fraudes (Classe = 1)	8.213	8.213
Melhor modelo	*Decision Tree*	*Decision Tree*
Acurácia	99,93%	99,69%
Recall	99,93%	99,69%
Precisão	99,92%	99,69%
F1 Score	99,92%	99,69%
Falsos Positivos	201 (0,01%)	0
Falsos Negativos	1.177 (0,06%)	15 (0,31%)
Tempo Treinamento	17,71s	9,19s
Tempo Predição	0,09s	0,08s
Tempo Total	17,80s	9,27s

Fonte: Elaborada pelo autor

Figura 36 - Matriz de confusão do modelo DT com dataset balanceado



Fonte: Elaborada pelo autor

5.11. TESTE DE CLASSIFICAÇÃO DO MELHOR MODELO

Para teste de classificação, foi selecionado um conjunto de atributos do dataframe original e aplicado no modelo de melhor performance, neste caso, Decision Tree (“modelo_dt”).

Com base nos resultados obtidos (Figura 37), podemos observar que o modelo DT prediziu corretamente o valor real da classe (“label = 0” e “prediction = 0”).

Figura 37 - Teste de classificação do melhor modelo

```
#valores para teste
entradas = [1.0, 9839.64,170136.0, 160296.36,0.0,0.0,2.0]
resultado = 0.0
```

```
#criando o dataframe Spark para execução do modelo
df_teste = spark.createDataFrame(
    [(resultado,\
      Vectors.dense([entradas[0],entradas[1],entradas[2],entradas[3],entradas[4],entradas[5],entradas[6]]))],
    ['label', 'features'])
```

```
#resultado da predição utilizando o modelo DT
resultado = model_dt.transform(df_teste)
resultado.select('label','prediction').show()
```

```
+-----+----------+
|label|prediction|
+-----+----------+
|  0.0|       0.0|
+-----+----------+
```

Fonte: Elaborada pelo autor

6. CONCLUSÕES E TRABALHOS FUTUROS

6.1. CONCLUSÕES

Os de algorimos de aprendizando de maquina introduzem elevado poder

computacional de tomada de decisão para sistemas de detecção de fraudes, conseguindo realizar predições de forma automatizada, rápida e assertida em frente às mudanças de padrões recorrentes no mercado finaceiro.

Então, podemos concluir que os sistemas que utilizam aprendizado de máquina possuem performance significativemente superior quando comparados com sistemas de regras genéricas e fixas que dependem da intervenção humana.

Ademais, a utilização do processamento distribuido contribui para o aumento da eficiência e redução do tempo de resposta nas etapas de mineração massiva dados, sendo imprescindível para o cenário de análise de movimentações financeiras.

Dentre os algoritmos avaliados, o modelo *Decision Tree* apresentou a melhor performance, tendo como relevante contribuição científica, além da alta acurácia, o reduzido tempo de predição de fraudes para volume massivo de transações, independentemente do nível de balanceamento do dataset.

6.2. TRABALHOS FUTUROS

Com base no conhecimento adquirido na presente pesquisa, vimos propor os trabalhos futuros relacionados a seguir:

- Testar e validar o modelo em bases de dados com transações PIX;
- Testar e validar o modelo em bases de dados reais;
- Utilizar outras técnicas de balanceamento, a exemplo do completamento para a classe majoritária;
- Incluir novos atributos para detecção de fraudes;
- Aplicar outras técnicas de balanceamento do dataset;
- Aplicar técnicas de otimização e parametrização dos algoritmos;
- Automatizar a importação do dataset via banco de dados SQL;
- Implantar o modelo em produção utilizando interface WEB ou mobile para entrada e saída de dados;
- Implantar o modelo em produção utilizando chatbots para entrada e saída de dados;
- Integrar sistemas multiagentes de IoT (Internet of Things).

Ademais, o estudo do mecanismo de classificação utilizando aprendizado de

máquina pode ser ampliado para outros domínios de aplicação, quais sejam:

- Segurança (predição de risco);
- Indústria (predição de falhas);
- Saúde (predição de diagnóstico de patologias);
- Educação (predição de evasão de alunos).

APÊNDICE

Link do notebook Python no Github: http://github.com/danielgleison/data-mining

REFERÊNCIAS

[1] ROJAS, Edgar Lopez. Synthetic Financial Datasets For Fraud Detection. Disponível: https://www.kaggle.com/ntnu-testimon/paysim1. Acesso em 24 out. 2020;

[2] OZA, Adity**.** Fraud Detection using Machine Learning. Disponível: http://cs229.stanford.edu/proj2018/report/261.pdf. Acesso em 24 out. 2020;

[3] ROJAS, Edgar Alonso Lopez. **Analysis of Fraud Controls Using the PaySim Financial Simulator.** Int. J. of Simulation and Process Modelling , 2017.

[4] RAMOS, José Abílio de Paiva. **Árvores de Decisão Aplicadas À Detecção de Fraudes Bancárias**. 2014. Dissertação (Mestrado Profissional em Computação Aplicada) – Universidade de Brasília, Brasília, 2014.

[5] JUNIOR, JOÃO CARLOS PACHECO. **Modelos para Detecção de Fraudes Utilizando Técnicas de Aprendizado de Máquina.** 2019. Dissertação (Mestrado Profissional em Economia) – Fundação Getúlio Vargas, São Paulo, 2019.

[6] JUNIOR, JOSÉ FELIPE. **Mineração de Dados para Detecção de Fraudes em Transações Eletrônicas**. 2012. Dissertação (Mestrado em Ciência da Computação). Universidade Federal de Minas Gerais. 2012.

[7] BRITO, João Guilherme Pinto Pedreira de. **Detecção de Fraude em Redes Financeiras com Modelação Baseada em Agentes.** 2018. Dissertação (Mestrado em Economia e Administração de Empresas) – Universidade do Porto, 2018.

[8] OLIVEIRA, Paulo Henrique Maestrello Assad. **Detecção de fraudes em cartões: um classificador baseado em regras de associação e regressão logística**. Dissertação (Mestrado em Ciência da Computação) – Universidade de São Paulo, São Paulo, 2016.

[9] JORDAN, M. I.; MITCHELL, T. M. **Machine learning: Trends, perspectives, and**

prospects. American Association for the Advancement of Science, v. 349, n. 6245, p. 255–260, 2015.

[10] AL-GARADI, M. A.; MOHAMED, A.; AL-ALI, A.; DU, X.; GUIZANI, M. **A survey of machine and deep learning methods for internet of things (iot) security**. arXiv preprint ar-Xiv:1807.11023, 2018.

[11] KOVACH, Stephan. **Deteção de Fraudes em Transações Financeiras via Internet em Tempo Real**. 2011. Tese (Doutorado em Engenharia Elétrica). Universidade de São Paulo. 2011.

[12] MENDES, Eduardo Cláudio Oliveira Mendes; LIMA Marcelo Campi. **Uso da inteligência artificial para identificação de fraudes financeiras: uma proposição de metodologias e ferramentas para detecção**. 2018. Monografia. Instituto de Educação Tecnológica. 2018.