Análise e Deteção de Fraudes em Transações de Cartões de Crédito dos Estados Unidos da América

Unidade Curricular de *Data Mining & Machine Learning*

Mestrado em Análise de Dados e Sistemas de Apoio à Decisão

Ano Letivo 2024 – 2025

Análise e Deteção de Fraudes em Transações de Cartões de Crédito dos Estados Unidos da América

Coimbra *Business School* | ISCAC

Coimbra, Portugal

Autores:

Bernardo Silva – 2020112296

Nuno Gonçalves – 2015063961

Simão Dias – 2020132169

Elaborado em

07/01/2025

Índice

Índice de Figuras

Lista de Siglas e Acrónimos

CRISP-DM: *Cross-Industry Standard Process for Data Mining*

CSV: *Comma-Separated Values*

EUA: Estados Unidos da América

GPU: *Graphics Processing Unit*

IBM: *International Business Machines Corporation*

ISCAC: Instituto Superior de Contabilidade e Administração de Coimbra

NY: Nova Iorque (*New York*)

RAM: *Random Access Memory*

TX: Texas

UNIX: Sistema Operativo UNI

OK: Oklahoma

IA: Iowa

PA: Pensilvânia

AK: Alasca

CT: Connecticut

ND: Dakota do Norte

1. Introdução

O seguinte projeto foi desenvolvido no âmbito da unidade curricular de **Data Mining** relativa ao **Mestrado em Análise de Dados e Sistemas de Apoio à Decisão** do **Instituto Superior de Contabilidade e Administração de Coimbra**. A estrutura do relatório está de acordo com a metodologia padrão **CRISP-DM** (*Cross – Industry Standard Process for Data Mining*), que identifica as diferentes fases na implementação de um projeto de mineração de dados (*Chapman et al*., 2000).

A deteção e análise de fraudes financeiras em transações de cartões de crédito representam um dos desafios mais importantes para as instituições financeiras no atual contexto global. As práticas fraudulentas comprometem a integridade financeira dos consumidores e das empresas, constituindo uma ameaça crescente à segurança económica. Com o avanço das tecnologias digitais e a proliferação do uso de cartões de crédito em transações comerciais, a complexidade e a sofisticação dos esquemas de fraude aumentaram significativamente.

Ao utilizarmos a metodologia CRISP-DM, reconhecida pela sua eficácia na estruturação de projetos analíticos, tem-se como principal objetivo explorar e preparar um conjunto de dados simulado que abrange transações legítimas fraudulentas ocorridas entre junho de 2020 e dezembro de 2020. O *dataset* utilizado foi gerado com recurso à ferramenta *Sparkov Data Generation*, que garante um cenário realista e reflete padrões de comportamento e irregularidades típicas em transações financeiras.

Com o projeto, pretende-se não só identificar e compreender os padrões associados a transações fraudulentas, mas também implementar e avaliar modelos preditivos que possam ser utilizados para mitigar riscos e prevenir fraudes futuras.

O projeto insere-se no contexto académico da unidade curricular, contribuindo para o desenvolvimento de competências práticas e analíticas relevantes para a análise de dados financeiros e a segurança digital.

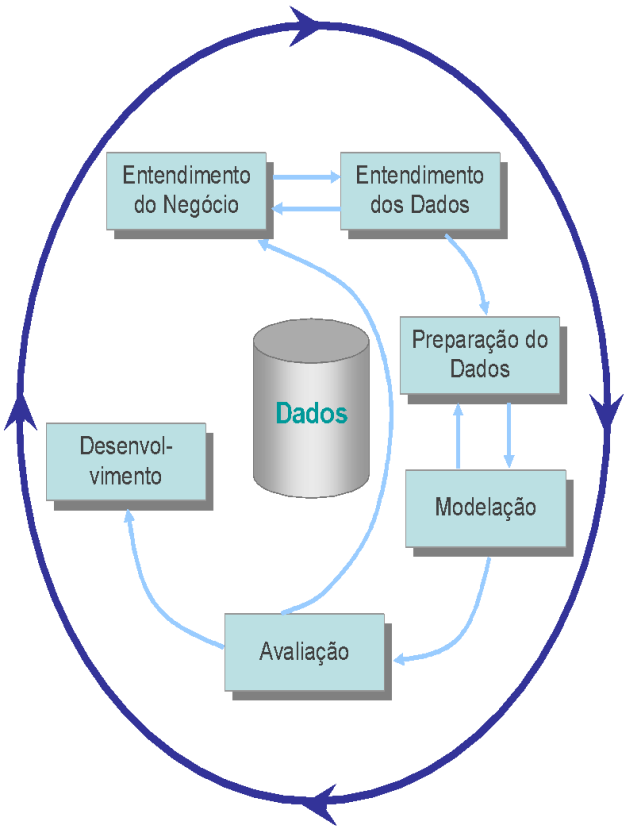


Figura 1: Fases do ciclo de vida da metodologia CRISP-DM

1.1. Entendimento do Tema

1.1.1. Evolução e Padrões das Fraudes em Cartões de Crédito nos Estados Unidos da América (EUA)

As fraudes em transações com cartões de crédito nos Estados Unidos têm apresentado uma tendência **crescente** nos últimos anos. De acordo com o Relatório Global de Tendências de Fraude Digital da *TransUnion*, entre 2019 e 2022, as transações digitais nos EUA aumentaram 89%, enquanto o volume de tentativas de fraude digital cresceu 122% no mesmo período.

Este crescimento é atribuído, em parte, à **proliferação de fraudes relacionadas à identidade**, como o roubo de identidade e a criação de identidades sintéticas. A fraude de identidade sintética, em particular, apresentou um aumento significativo, com um crescimento de 76% entre 2019 e 2022.

Além disso, um estudo da IBM revelou que os **cidadãos americanos** são as **vítimas mais frequentes de fraudes em cartões de crédito** em comparação com outros países. Os utilizadores de cartões de crédito nos EUA relataram a

segunda maior quantidade de dinheiro perdido em transações fraudulentas nos últimos 12 meses, ficando apenas atrás do Japão.

A análise por gerações indica que os *Millennials* (nascidos entre 1981 e 1996) são consistentemente as maiores vítimas de todas as formas de fraude financeira. A Geração X (nascidos entre 1965 e 1980) reportou o segundo maior número de cobranças fraudulentas, seguida pela Geração Z (nascidos entre 1997 e 2012).

Em termos de volume de contas, o número de contas de cartão de crédito nos EUA tem aumentado ao longo dos anos, passando de 377,90 milhões em 2003 para 599,10 milhões em junho de 2024, segundo o *Trading Economics*. Este aumento no número de contas está correlacionado com o crescimento das transações digitais e, consequentemente, com a exposição a potenciais fraudes.

O seguinte gráfico ilustra a **evolução do número de fraudes em cartões de crédito nos Estados Unidos** entre 2015 e 2022.

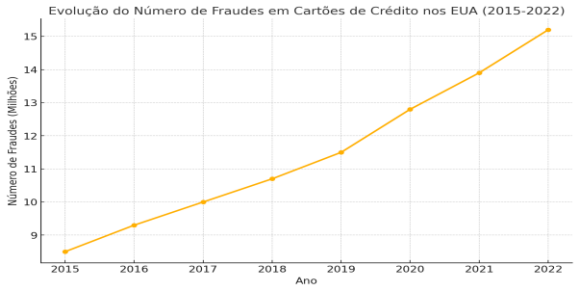


Figura 2: Evolução do número de fraudes em cartões de crédito nos EUA (2015-2022)

1.1.2. Avaliação da Situação Atual

Numa perspetiva de caracterização detalhada dos recursos necessários para este projeto, considera-se essencial identificar os requisitos em termos de recursos humanos, materiais e tecnológicos.

Recursos Humanos:

O projeto será desenvolvido, exclusivamente, pelos três estudantes do curso de Mestrado em Análise de Dados e Sistemas de Apoio à Decisão, que irão desempenhar todas as funções relacionadas com a limpeza, preparação e modelação dos dados, bem como a análise e interpretação dos resultados.

Recursos Materiais:

- **Base de dados**: O conjunto de dados utilizado está no formato CSV, ideal para a análise proposta, devido à sua compatibilidade e eficiência com as ferramentas escolhidas.

- **Hardware**: A execução do projeto não exige equipamento de alto desempenho. As especificações recomendadas incluem:

- **Processador**: AMD *Ryzen* 5 5600X ou equivalente, para garantir um bom desempenho.
- **Memória *RAM***: 16 GB, suficiente para lidar com os dados e os cálculos.
- **Armazenamento**: 500 GB *SSD*, para assegurar rapidez no acesso aos dados e às ferramentas.
- **Placa gráfica**: Não é necessária uma *GPU* avançada; qualquer placa padrão é suficiente.
- **Software**: Serão utilizadas ferramentas e bibliotecas amplamente disponíveis e gratuitas na linguagem *Python*, como *Pandas*, *NumPy*, *Matplotlib*, *Scikit-learn*. O sistema operativo *Windows* 10/11 será suficiente para a execução.

Disponibilidade:

Os três estudantes estão comprometidos com o projeto e vão dividir o trabalho entre a análise dos dados, a sua preparação, a construção de modelos e a documentação. Estima-se que a dedicação ao projeto seja realizada em horários flexíveis durante a semana e aos fins de semana.

Custo-benefício:

O projeto não apresenta qualquer tipo de custos financeiros, uma vez que será realizado integralmente pelos estudantes, que vão utilizar os recursos próprios. Dado o impacto potencial deste estudo na deteção e prevenção de fraudes financeiras, o conhecimento gerado será uma mais-valia, tanto para fins académicos quanto para a aplicação prática em cenários reais.

1.2. Definição dos Objetivos do *Data Mining*

Como referido anteriormente, o principal objetivo do projeto é identificar possíveis padrões de fraude em transações de cartões de crédito e desenvolver um modelo preditivo eficaz que permita diferenciar transações legítimas de fraudulentas. O estudo baseia-se num conjunto de dados que contém transações realizadas entre junho de 2020 e dezembro de 2020.

O foco está em construir um modelo preditivo capaz de:

- **Identificar transações fraudulentas** com elevada precisão, minimizando falsos positivos e falsos negativos.
- **Reduzir riscos financeiros** associados a fraudes, fornecendo insights que possam ser usados por instituições financeiras para fortalecer os seus sistemas de segurança.

- **Promover a eficiência operacional**, ao automatizar o processo de deteção de fraudes.

A modelação será realizada utilizando a linguagem *Python*, com o apoio de bibliotecas como *Pandas*, *NumPy*, *Matplotlib*. Estas ferramentas serão fundamentais para a análise, visualização e avaliação dos dados, e para garantir que o modelo preditivo responde aos objetivos definidos.

Através dos resultados obtidos pelo modelo, pretende-se avaliar o seu desempenho com base em métricas como precisão, *recall* e taxa de falsos positivos. Será realizada uma análise crítica dos resultados, considerando as suas limitações e potenciais melhorias.

Este trabalho, atende aos objetivos académicos da unidade curricular, mas também demonstra a aplicabilidade de técnicas de mineração de dados na mitigação de fraudes financeiras, sublinhando o valor prático da análise preditiva para o setor.

1.3. Produzir o Plano do Projeto

O plano do projeto deve ser estruturado de forma a minimizar riscos e evitar possíveis falhas ao longo das etapas. Para tal, foi definido um conjunto de passos que guiarão a execução do trabalho, com uma duração estimada de dois meses.

O projeto será organizado em cinco fases principais:

1. **Preparação dos dados:**

Nesta fase inicial, serão analisados os dados para identificar possíveis problemas como valores nulos ou inconsistências. Em seguida, será realizada a limpeza dos dados, eliminando ou imputando valores ausentes (N/A) de modo a garantir a sua integridade e qualidade para as etapas seguintes.

2. **Seleção de Atributos:**

Será realizada uma análise exploratória para determinar as colunas do *dataset* que não contribuem significativamente para a análise. Estas colunas serão removidas, de forma a garantir que vão ser mantidos apenas os atributos relevantes para a deteção de fraudes.

3. **Modelação:**

Será aplicado um modelo preditivo baseado em técnicas de *Machine Learning*, utilizando a linguagem *Python* e bibliotecas como *Scikit-learn*.

4. **Avaliação e Conclusões:**

Nesta última fase, os resultados do modelo vão ser analisados criticamente, avaliando métricas como a precisão, *recall* e taxa de falsos positivos. Irão ser retiradas conclusões sobre o desempenho do modelo e propostas melhorias que possam ser implementadas em estudos futuros.

Este plano permite uma abordagem estruturada e eficiente de modo a assegurar que todas as etapas essenciais sejam devidamente consideradas.

2. Estudo dos Dados

2.1. Recolha dos Dados Iniciais

Os dados utilizados neste projeto foram recolhidos através da plataforma *Kaggle*, especificamente no conjunto de dados disponível em:

https://www.kaggle.com/datasets/kartik2112/fraud-detection/data

Este *dataset* foi selecionado por ser uma **simulação realista de transações de cartões de crédito**, que contém tanto transações legítimas como fraudulentas.

Tal como especificamos anteriormente, este conjunto de dados foi gerado pela ferramenta *Sparkov Data Generation*, que permite a criação de dados simulados com base em padrões realistas de comportamento financeiro.

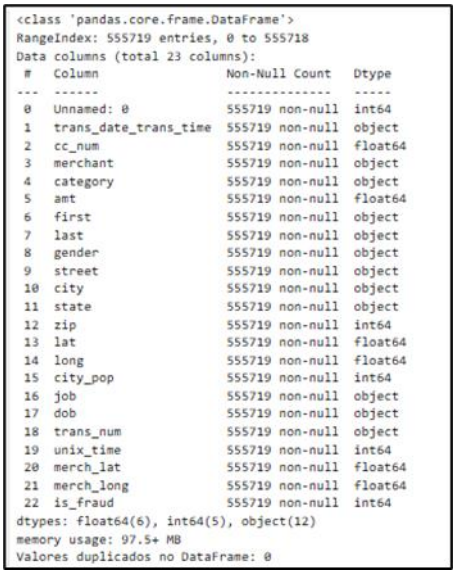
Após o download, os dados foram fornecidos em formato CSV, que facilita a sua importação e manipulação ao utilizar a ferramenta *Python*. Antes de avançar para a análise, o *dataset* foi examinado para garantir a sua integridade e qualidade, onde se identificou valores ausentes e/ou inconsistências.

Esta fonte de dados é essencial para o desenvolvimento do modelo preditivo, uma vez que fornece uma base rica em informações que refletem padrões comuns em fraudes financeiras.

2.2. Descrição dos Dados

O *dataset* utilizado neste projeto contém um total de **555.719 registos e 23 variáveis**. Cada registo representa uma transação de cartão de crédito, que inclui tanto transações legítimas como fraudulentas. Numa primeira análise, verificamos também que não existem valores nulos nem duplicados no dataset.

As variáveis disponíveis no *dataset* estão descritas em seguida:



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 555719 entries, 0 to 555718
Data columns (total 23 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   Unnamed: 0             555719 non-null  int64
 1   trans_date_trans_time  555719 non-null  object
 2   cc_num                 555719 non-null  float64
 3   merchant               555719 non-null  object
 4   category               555719 non-null  object
 5   amt                    555719 non-null  float64
 6   first                  555719 non-null  object
 7   last                   555719 non-null  object
 8   gender                 555719 non-null  object
 9   street                 555719 non-null  object
 10  city                   555719 non-null  object
 11  state                  555719 non-null  object
 12  zip                    555719 non-null  int64
 13  lat                    555719 non-null  float64
 14  long                   555719 non-null  float64
 15  city_pop               555719 non-null  int64
 16  job                    555719 non-null  object
 17  dob                    555719 non-null  object
 18  trans_num              555719 non-null  object
 19  unix_time              555719 non-null  int64
 20  merch_lat              555719 non-null  float64
 21  merch_long             555719 non-null  float64
 22  is_fraud               555719 non-null  int64
dtypes: float64(6), int64(5), object(12)
memory usage: 97.5+ MB
Valores duplicados no DataFrame: 0
```

Figura 3: Variáveis disponíveis no Dataset

- **Unnamed:** 0: ID de cada coluna.
- **trans_date_trans_time:** Data e hora da transição;
- **cc_num:** Número do cartão de crédito (anonimizado);
- **merchant:** Nome do comerciante;
- **category:** Categoria da transação (por exemplo, "*personal_care*");
- **amt:** Valor da transação;
- **first e last:** Nome e apelido do titular do cartão;
- **gender:** Género do titular do cartão;
- **street, city, state, zip:** Informações de endereço do titular;
- **lat e long:** Coordenadas geográficas do titular do cartão;
- **city_pop:** População da cidade onde a transação foi efetuada;
- **job:** Profissão do titular do cartão;
- **dob:** Data de nascimento do titular;
- **trans_num:** Identificador único da transação;
- **unix_time:** *Timestamp* da transação;
- **merch_lat e merch_long:** Coordenadas do local do comerciante;
- **is_fraud:** Indicador binário de fraude (0 para legítima, 1 para fraudulenta).

O **ID de cada linha já está implícito no *Python***, uma vez que cada registo possui um índice automático (conforme demonstrado na figura anterior). Desta forma, a coluna presente no *dataset*, "***Unnamed: 0***", é redundante, ou seja, não

adiciona qualquer valor à análise. Além disso, a coluna foi identificada automaticamente como "*Unnamed: 0*" pelo *Python*, devido à ausência de um título original no ficheiro. Este campo será removido posteriormente, uma vez que não apresenta utilidade prática.

De seguida apresentamos a função *df.describe*:

	Unnamed: 0	cc_num	amt	zip	lat	long	city_pop	unix_time	merch_lat	merch_long	is_fraud
count	555719.000000	5.557190e+05	555719.000000	555719.000000	555719.000000	555719.000000	5.557190e+05	5.557190e+05	555719.000000	555719.000000	555719.000000
mean	277859.000000	4.178387e+17	69.392810	48842.628015	38.543253	-90.231325	8.822189e+04	1.380679e+09	38.542798	-90.231380	0.003860
std	160422.401459	1.309837e+18	156.745941	26855.283328	5.061336	13.721780	3.003909e+05	5.201104e+06	5.095829	13.733071	0.062008
min	0.000000	6.041621e+10	1.000000	1257.000000	20.027100	-165.672300	2.300000e+01	1.371817e+09	19.027422	-166.671575	0.000000
25%	138929.500000	1.800430e+14	9.630000	26292.000000	34.668900	-96.798000	7.410000e+02	1.376029e+09	34.755302	-96.905129	0.000000
50%	277859.000000	3.521420e+15	47.290000	48174.000000	39.371600	-87.476900	2.408000e+03	1.380762e+09	39.376593	-87.445204	0.000000
75%	416788.500000	4.635330e+15	83.010000	72011.000000	41.894800	-80.175200	1.968500e+04	1.385867e+09	41.954163	-80.264637	0.000000
max	555718.000000	4.992350e+18	22768.110000	99921.000000	65.689900	-67.950300	2.906700e+06	1.388534e+09	66.679297	-66.952026	1.000000

Figura 4: Visualização da variável describe em detalhe

Ao analisarmos a função, *df_describe*, conseguimos concluir o seguinte:

Distribuição dos Dados:

- Os valores das transações variam entre **1,00$ e 22.768,11$**, com uma média de aproximadamente **69,39$**;
- A população média (*city_pop*) é de **88.221 habitantes** e um máximo de **2.906.700 habitantes**, o que sugere que as transações ocorrem tanto em grandes centros urbanos como em áreas menores;
- Apenas **0,386%** das transações são marcadas como **fraudulentas** (*is_fraud* = 1).

Como foi possível verificar acima, uma das variáveis que está presente no nosso dataset, é a variável *"trans_date_trans_time"*, que contém a data e a hora da transação.

No entanto, esta informação encontra-se toda junta. Sendo uma variável do tipo categórica, achamos por bem em transformar esta variável em *datetime* e separar esta coluna para hora, dia da semana e mês, uma vez que assumimos que essa informação pode ser importante para o projeto, permitindo ter uma melhor visão e **analisar melhor os âmbitos comportamentais das transações fraudulentas**.

Posto isto, aplicamos os códigos que seguem e conseguimos obter novas colunas/variáveis correspondentes ao pretendido.

```
#converter trans_date_trans_time para o formato date
df['trans_date_trans_time'] = pd.to_datetime(df['trans_date_trans_time'])
print(df.dtypes['trans_date_trans_time'])
df.head()
```

```
datetime64[ns]
```

Figura 5: Conversão trans_date_time para o formato date

```
# derivar colunas adicionais a partir de 'trans_date_trans_time'
#derivar a hora
df['trans_hour'] = df['trans_date_trans_time'].dt.hour
#derivar o 'dia da semana'
df['trans_day_of_week'] = df['trans_date_trans_time'].dt.day_name()
#derivar o 'ano_mês'
df['trans_year_month'] = df['trans_date_trans_time'].dt.to_period('M')
```

Figura 6: Derivar colunas adicionais a partir de 'trans_date_trans_time'

Ainda relativamente à questão das datas, verificamos também que o *dataset* tem a coluna "*dob*" que corresponde a data de nascimento do titular do cartão de crédito, e achamos por bem derivar desta variável a idade do titular, criando uma nova variável "*age"*, para que nos possa dar *insights* valiosos no sentido de percebermos a **faixa etária das pessoas que mais sofreram com fraude em cartão de crédito**.

Assim sendo, aplicamos o código e apresentamos de seguida as estatísticas descritivas da variável *"age"*.

```
# Certificar-se de que as colunas estão no formato datetime
df['dob'] = pd.to_datetime(df['dob'])
df['trans_date_trans_time'] = pd.to_datetime(df['trans_date_trans_time'])

# Calcular a idade em anos
df['age'] = (df['trans_date_trans_time'] - df['dob']).dt.days // 365

# Exibir as 5 primeiras linhas da nova coluna 'age'
print(df['age'].head())
```

Figura 7: Nova coluna 'age'

```
count   555719.000000
mean        46.423797
std         17.442737
min         15.000000
25%         33.000000
50%         44.000000
75%         58.000000
max         96.000000
Name: age, dtype: float64
```

Figura 8: Estatísticas descritivas da variável "age".

Uma vez que extraímos variáveis novas da coluna *"trans_date_trans_time"*, e da coluna *"dob"*, decidimos dar *drop* nas mesmas, uma vez que não nos irão trazer mais utilidade neste contexto.

Ainda relativamente ao *drop* de variáveis, decidimos também dar *drop* nas variáveis *"first"* e *"last"*, que correspondem ao primeiro e último nome, respetivamente do titular do cartão de crédito, uma vez que estas variáveis podem levantar questões ao nível da privacidade dos dados e também não possuem qualquer relevância para o contexto do nosso projeto.

2.3. Exploração dos Dados

De seguida, decidimos testar a correlação existente entre os dados, para uma primeira avaliação, como demonstra a Figura 9.

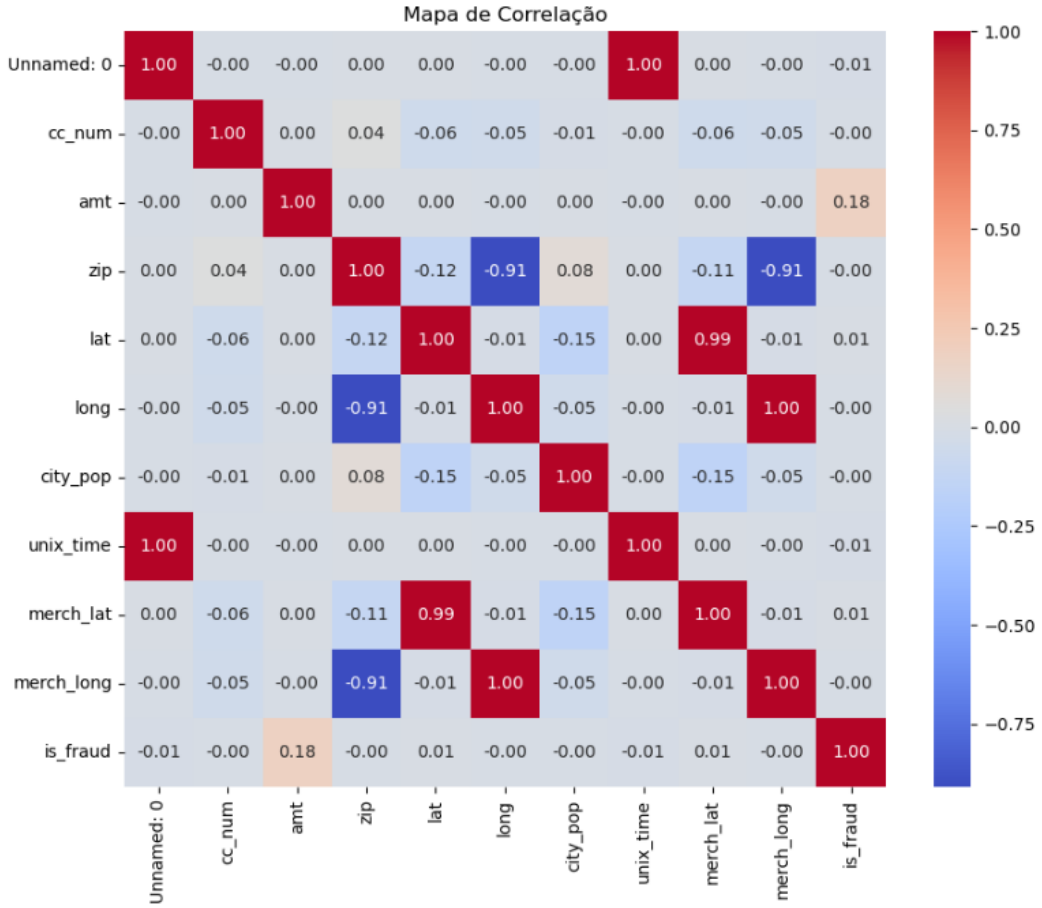


Figura 9: Mapa de Correlação entre variáveis

A seguinte demonstração serve para compreender melhor as nossas variáveis e como se relacionam entre si.

A partir desta análise, pode-se concluir o seguinte:

Variáveis com Correlação Forte (próximas de 1 ou -1):

- **lat e merch_lat (0.99):**

Existe uma **correlação positiva muito forte** entre (*lat*) e a latitude da loja (*merch_lat*).

Indica que essas variáveis representam informações geográficas muito próximas ou redundantes.

- **long e merch_long (1.00):**

A **correlação perfeita** entre longitude (*long*) e longitude da loja (*merch_long*) também sugere redundância.

Indica que ambas as variáveis têm valores idênticos ou extremamente próximos.

- ***Unnamed:* 0**

Esta variável não tem nenhuma relevância com outras variáveis, uma vez que não apresenta nenhuma correlação relevante com qualquer outra variável.

Tal como tínhamos verificado anteriormente, é um ID duplicado.

Passo seguinte: **Apagar a variável.**

Após analisar as correlações entre as variáveis presentes no conjunto de dados, partimos para uma exploração mais visual e descritiva, visando identificar padrões comportamentais, demográficos e temporais. A análise gráfica apresentada permite compreender melhor os fatores que influenciam as fraudes, assim como os contextos em que estas são mais propensas a ocorrer.

A variável *is_fraud* é a nossa variável alvo, representando a classificação das transações como legítimas (valor 0) ou fraudulentas (valor 1). Esta variável é de extrema importância no projeto, uma vez que o objetivo principal é desenvolver um modelo eficaz para a deteção de fraudes em transações de cartões de crédito.

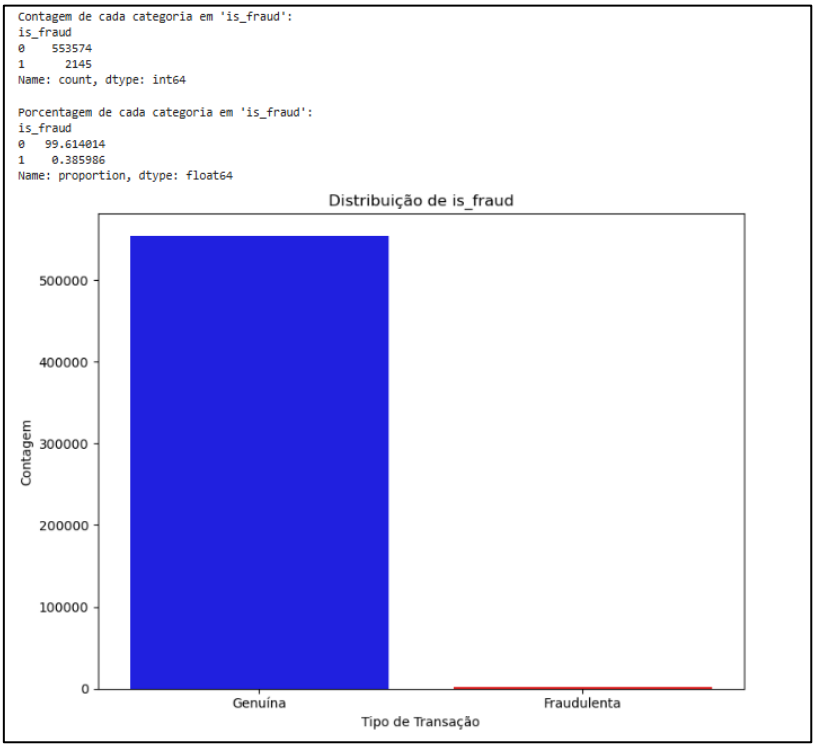


Figura 10: Distribuição 'is_fraud'

Contagem de cada categoria:

- Transações genuínas (legítimas): 553,574 (99.61%)
- Transações fraudulentas: 2,145 (0.39%)

Proporção relativa:

A esmagadora maioria das transações pertence à categoria de transações genuínas, representando mais de 99.6% do total.

Apenas 0.39% das transações são classificadas como fraudulentas.

O gráfico acima ilustra esta **disparidade** de forma clara, onde a barra azul (transações genuínas) é substancialmente maior do que a barra vermelha (transações fraudulentas).

Esta **distribuição desbalanceada apresenta vários desafios importantes** para o desenvolvimento de modelos preditivos:

1. **Domínio da Classe Maioritária:**

Dado o elevado número de transações genuínas, um modelo simples poderia atingir uma elevada precisão (*accuracy*) ao classificar todas as transações como genuínas. No entanto, seria um resultado enganador, uma vez que que as fraudes, **apesar de raras, são as mais relevantes para o estudo.**

2. **Desempenho nas Métricas de Fraudes:**

Métricas como *Recall*, *F1-Score* e AUC tornam-se particularmente relevantes, pois avaliam a capacidade do modelo em identificar as fraudes sem comprometer demasiado a classificação de transações legítimas.

3. **Impactos Práticos**:

A falha em identificar fraudes (*False Negatives*) pode gerar perdas significativas para o banco, enquanto a classificação incorreta de transações legítimas como fraudulentas (*False Positives*) pode afetar a experiência do cliente.

4. **Soluções Planeadas**:

Dado este desbalanceamento, é crucial aplicar técnicas de manipulação de dados que ajustem a proporção entre as classes para melhorar a capacidade do modelo de aprender e detetar fraudes. O projeto irá abordar esta questão através de **Técnicas de Balanceamento**, métodos como ***Oversampling*** e ***Undersampling*** serão explorados para lidar com o **desbalanceamento**.

Estratégias de Avaliação: Métricas ajustadas para classes desbalanceadas serão priorizadas na análise do desempenho dos modelos.

Esta abordagem será apresentada e detalhada em capítulos posteriores, onde os métodos de balanceamento serão aplicados e os seus impactos no desempenho dos modelos serão avaliados.

Análise Variáveis temporais:

O gráfico apresenta a evolução do número de transações ao longo do tempo, agrupadas por ano e mês. Observa-se um aumento inicial significativo nas transações de 2020-06 para 2020-07, seguido de uma ligeira queda e estabilização nos meses subsequentes. Contudo, há um crescimento acentuado no mês de 2020-12, sugerindo um pico sazonal, possivelmente relacionado ao aumento de compras durante o período festivo.

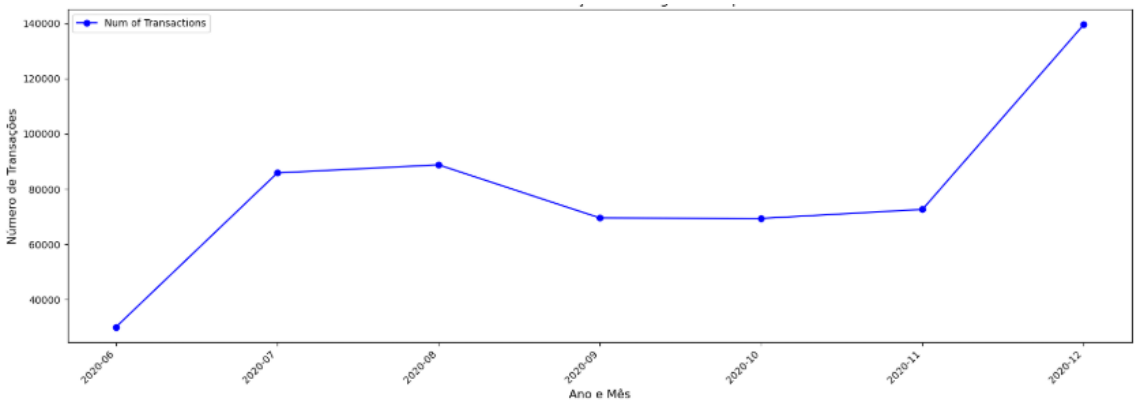


Figura 11: Número de Transações ao Longo do Tempo

Número de Fraudes por Dia da Semana:

O gráfico mostra a distribuição de fraudes ao longo dos dias da semana. Observa-se que os dias de terça-feira e domingo registam o maior número de fraudes. Esta distribuição pode indicar uma preferência por realizar transações em dias específicos, possivelmente quando os sistemas de monitorização são menos eficazes.

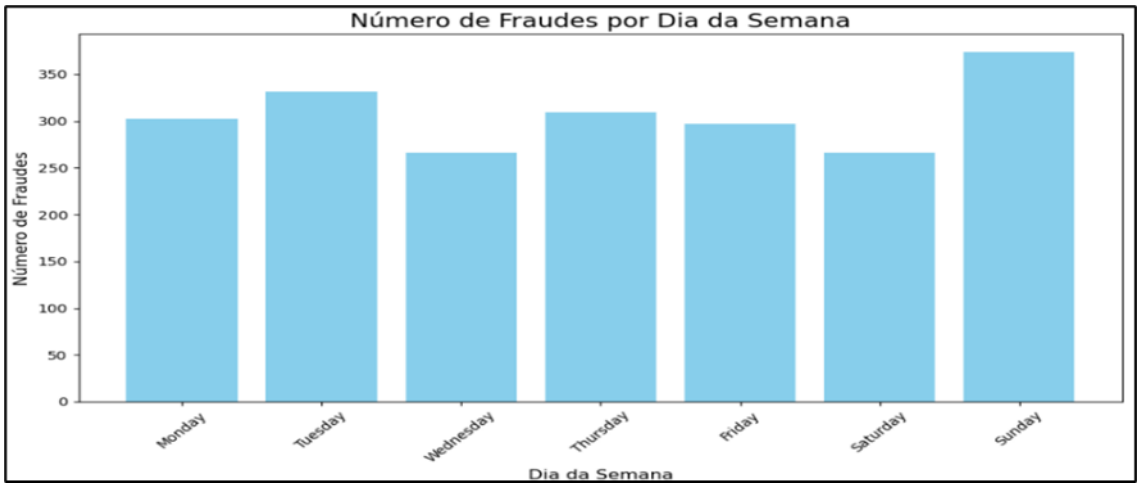
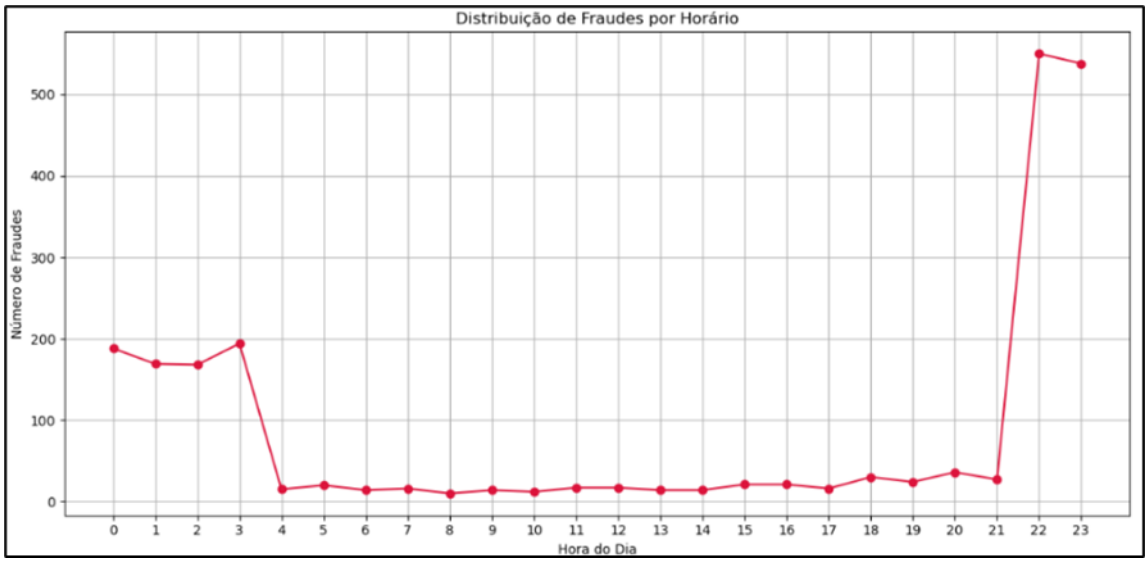


Figura 12: Número de fraudes por dia de semana

Distribuição de Fraudes por Horário:

O gráfico apresenta a distribuição de fraudes ao longo das horas. Nota-se que os períodos da meia-noite até às 3 da manhã e entre as 22h e 23h concentram a maior quantidade de fraudes. Estas horas correspondem a períodos de menor atividade humana, o que pode ser explorado para evitar deteção. Durante o restante do dia, o número de fraudes mantém-se baixo, mas relativamente constante.



Fraudes por Período do Dia:

O gráfico categoriza as fraudes em quatro períodos do dia: Manhã, Tarde, Noite e Madrugada. A noite concentra o maior número de fraudes, seguida pela madrugada. A tarde e a manhã registam números significativamente mais baixos, reforçando a ideia de que quem comete fraudes prefere períodos de menor vigilância ou atividade.

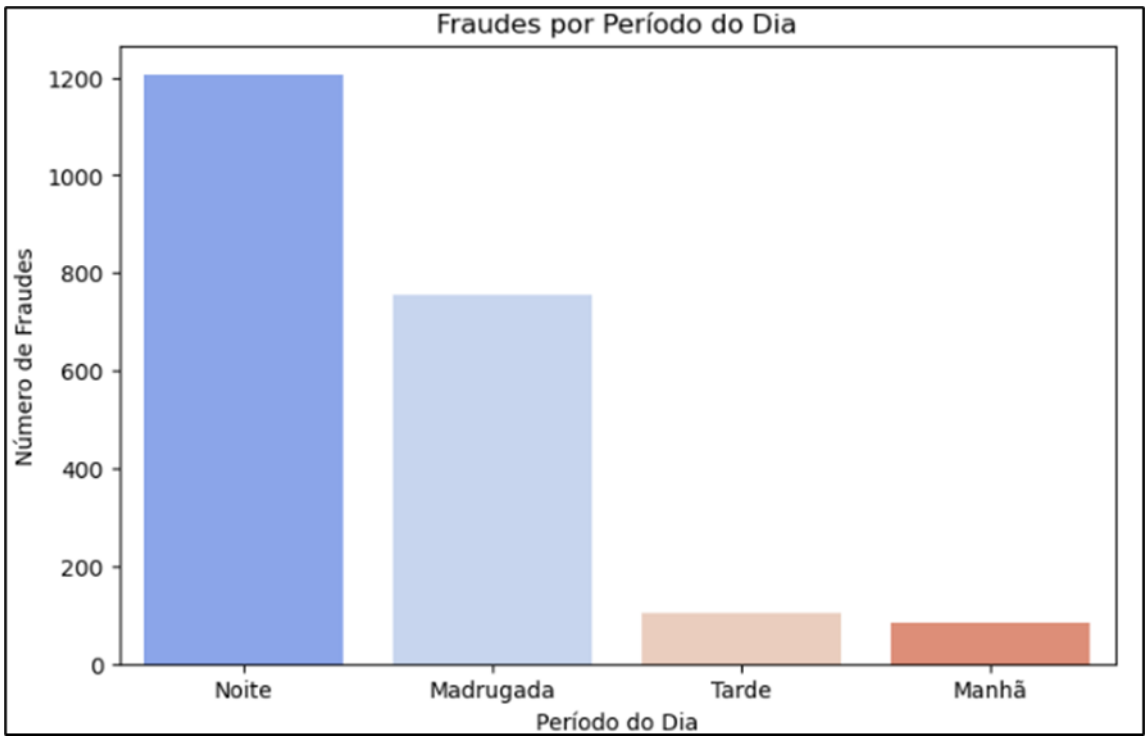


Figura 13: Fraudes por período do dia

Análise Variável *"amt"*:

O objetivo desta análise é compreender o comportamento dos valores transacionados, dividindo-os entre transações gerais, genuínas e fraudulentas.

Assim sendo, apresenta-se uma tabela onde são fornecidas as estatísticas descritivas para cada tipo de transação, permitindo identificar padrões e diferenças entre os vários tipos de transação.

	Row Type	Distribuição geral de amt	Distribuição geral de amt em transações genuínas	Distribuição geral de amt em transações fraudulentas
0	count	555719.000000	553574.000000	2145.000000
1	mean	69.392810	67.614408	528.356494
2	std	156.745941	152.471931	392.747594
3	min	1.000000	1.000000	1.780000
4	50%	47.290000	47.150000	371.940000
5	95%	193.051000	188.870000	1084.108000
6	99.9%	1572.723500	1575.960200	1311.443760
7	max	22768.110000	22768.110000	1320.920000

Figura 14: Estatísticas descritivas para cada tipo de transação

A tabela apresenta as estatísticas descritivas dos valores transacionados *'amt'* para três categorias: transações gerais, genuínas e fraudulentas. As principais conclusões são as seguintes:

1. **Transações Gerais:**

- O número total de transações é 555.719, com um valor médio de 69,39 e uma mediana de 47,29.
- A maioria das transações concentra-se em valores baixos, como evidenciado pelo percentil de 95% (193,05). No entanto, há valores atípicos, com o máximo a atingir os 22.786,11.

2. **Transações Genuínas:**

- Representam a grande maioria das transações, totalizando 553.574 (mais de 99% do total).
- A média (67,61) e a mediana (47,15) são muito próximas às transações gerais, isso indica que dominam o comportamento global.
- Os valores genuínos também incluem os valores mais elevados (máximo de 22.786,11), sugerindo que os *outliers* são maioritariamente associados a transações genuínas.

3. **Transações Fraudulentas:**

- Apenas 2.145 transações (cerca de 0,4% do total) foram identificadas como fraudulentas, demonstrando a sua raridade.
- No entanto, a média (528,36) é significativamente superior às transações gerais e genuínas, indicando que fraudes tendem a ocorrer com valores mais elevados.
- A mediana (371,94) e o percentil de 95% (1.084,10) confirmam que estas transações estão concentradas em faixas de valores superiores.
- O valor máximo (1.320,92) das transações fraudulentas é mais baixo do que o máximo das genuínas, sugerindo que fraudes evitam valores extremamente altos para passar despercebidas.

Com o objetivo de complementar a análise estatística apresentada anteriormente, os gráficos fornecem uma visualização clara da distribuição dos valores transacionados, permitindo identificar padrões distintos entre transações gerais, genuínas e fraudulentas. São apresentados um *boxplot*, que evidencia a dispersão e os *outliers*, e histogramas que ilustram a frequência das transações em diferentes faixas de valores.

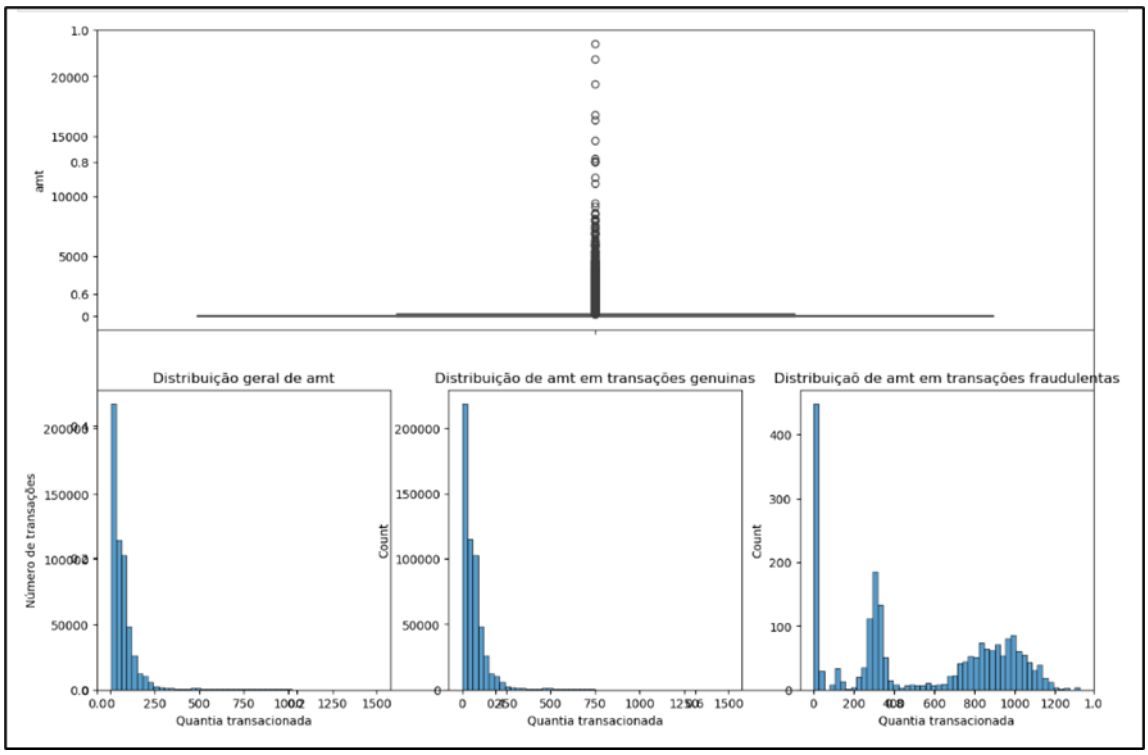


Figura 15: Visualizações referentes à variável 'amt'

Os gráficos apresentados evidenciam os seguintes pontos fundamentais:

1. *Boxplot* da Distribuição Geral:

- A maioria das transações ocorre em valores baixos, abaixo de 200$.
- A presença de *outliers* significativos (valores acima de 1.500) demonstra a existência de transações atípicas que podem impactar a análise geral.

2. Histogramas:

- **Distribuição Geral e Genuína:** Ambas mostram uma concentração elevada de transações de baixos valores, com uma diminuição rápida à medida que os valores aumentam.
- **Distribuição Fraudulenta:** O padrão é visivelmente distinto, com a maioria das fraudes concentradas em valores médios (entre 400 e 800). Há uma ausência quase completa de fraudes em valores baixos (<200), o que as diferencia das transações genuínas.

Estes gráficos confirmam as tendências identificadas na tabela, com fraudes a destacarem-se em faixas de valores específicas, enquanto as genuínas dominam as transações de valores mais baixos.

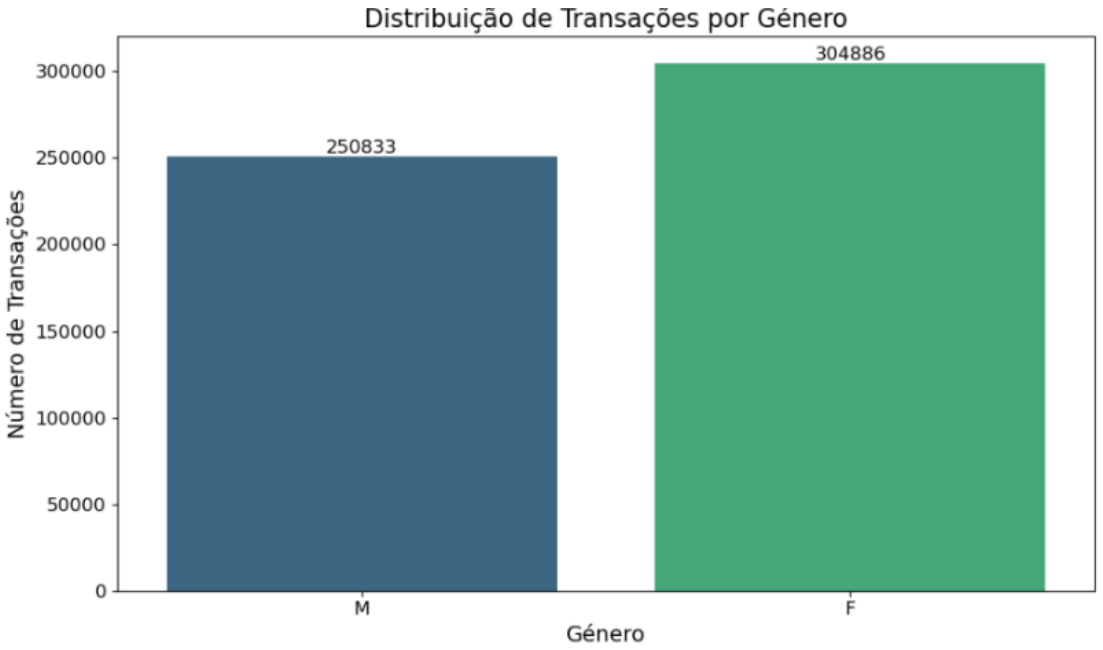


Figura 16: Distribuição de transações por género

A análise da variável *gender* é crucial para compreender possíveis diferenças no comportamento transacional entre géneros. Esta variável pode fornecer *insights* valiosos sobre padrões associados a transações, permitindo identificar se existe uma relação entre o género e a ocorrência de fraudes.

De seguida, analisamos os dados para explorar essas possíveis relações. Para isso, vamos apresentar alguns gráficos e vamos avançar com a análise:

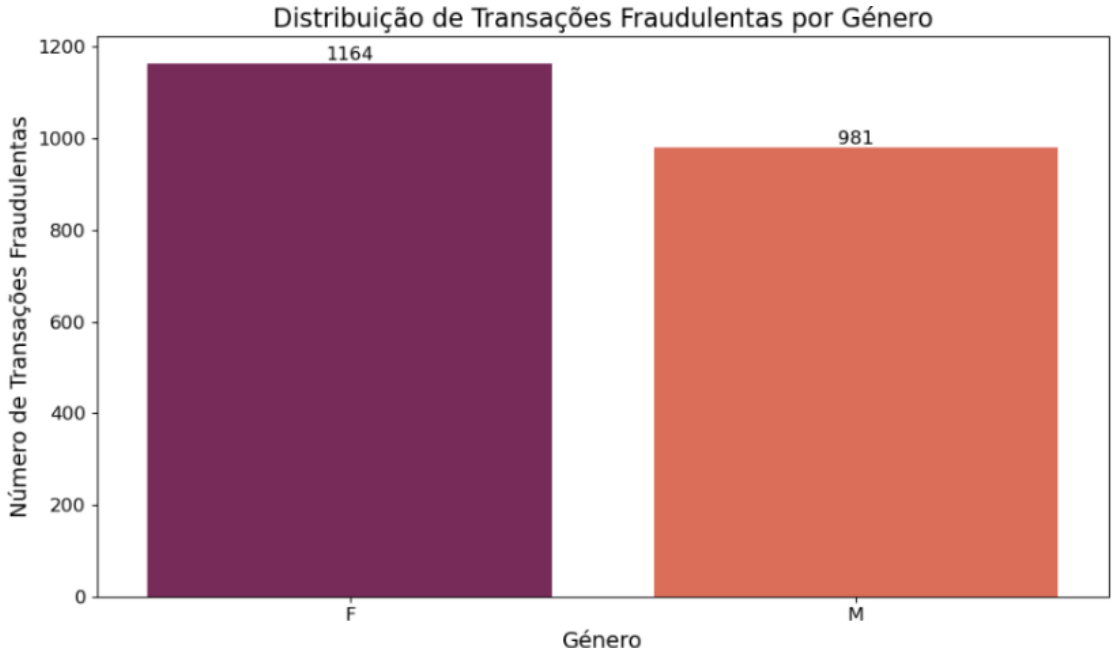


Figura 17: Percentual de transações fraudulentas por género

- **Distribuição de Transações por Género:**

Os dados mostram que as mulheres realizam um número maior de transações do que os homens, com 304.886 transações femininas contra 250.833 masculinas. Este padrão pode refletir diferenças no comportamento de consumo, com as mulheres possivelmente a utilizarem os seus meios de pagamento de forma mais frequente.

- **Distribuição de Transações Fraudulentas por Género:**

Entre as transações fraudulentas, as mulheres também apresentam um número maior de casos, com 1.164 fraudes em comparação com 981 fraudes nos homens. Embora o número total de fraudes seja superior para as mulheres, a diferença é proporcionalmente menor quando comparada ao volume total de transações realizadas por cada género. Isto sugere que a frequência de fraudes entre géneros é relativamente equilibrada.

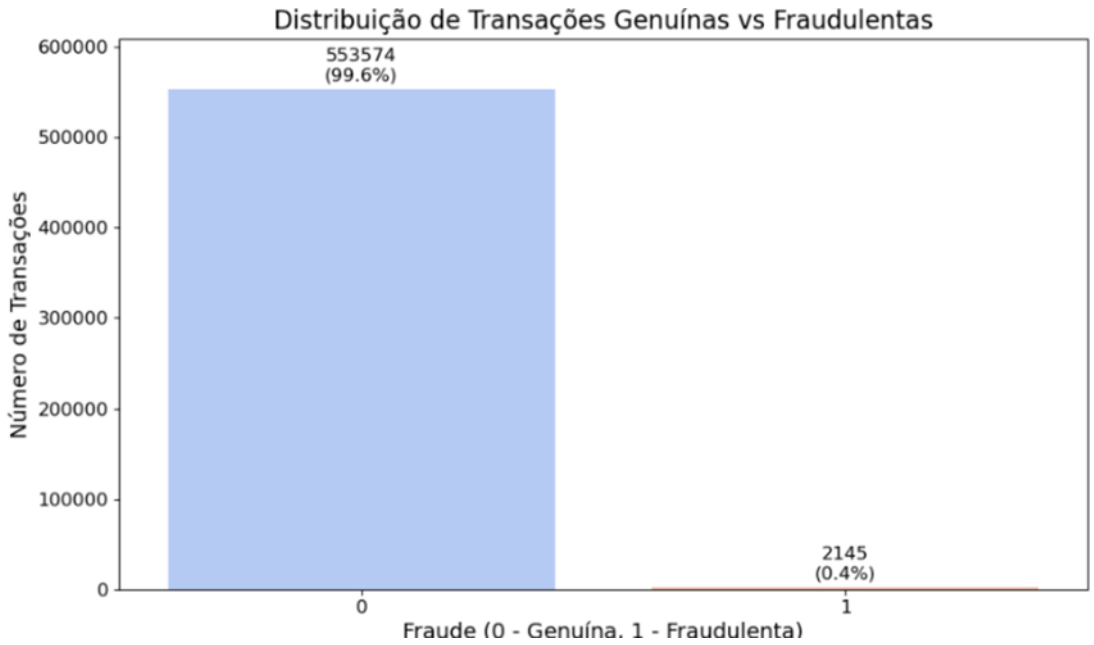


Figura 18: Distribuição de transações genuínas vs fraudulentas

A proporção de transações fraudulentas é extremamente baixa (0,4%) em comparação com as genuínas (99,6%). Este desbalanceamento indica que existe necessidade de utilizar técnicas de balanceamento das classes.

	year_month	num_of_fraud_transactions	fraud_customers
0	2020-06	133	15
1	2020-07	321	35
2	2020-08	415	41
3	2020-09	340	35
4	2020-10	384	39
5	2020-11	294	31
6	2020-12	258	26

Figura 19: Número de transações fraudulentas vs clientes fraude

Análise variável "*Category*":

Esta análise é essencial para compreender como é que os diferentes tipos de despesas se distribuem. Este tipo de análise permite identificar padrões de comportamento que podem ser úteis para reforçar a monitorização de determinadas categorias mais suscetíveis a fraudes. A seguir, analisamos a distribuição geral das transações por categoria e a proporção de fraudes em cada uma delas.

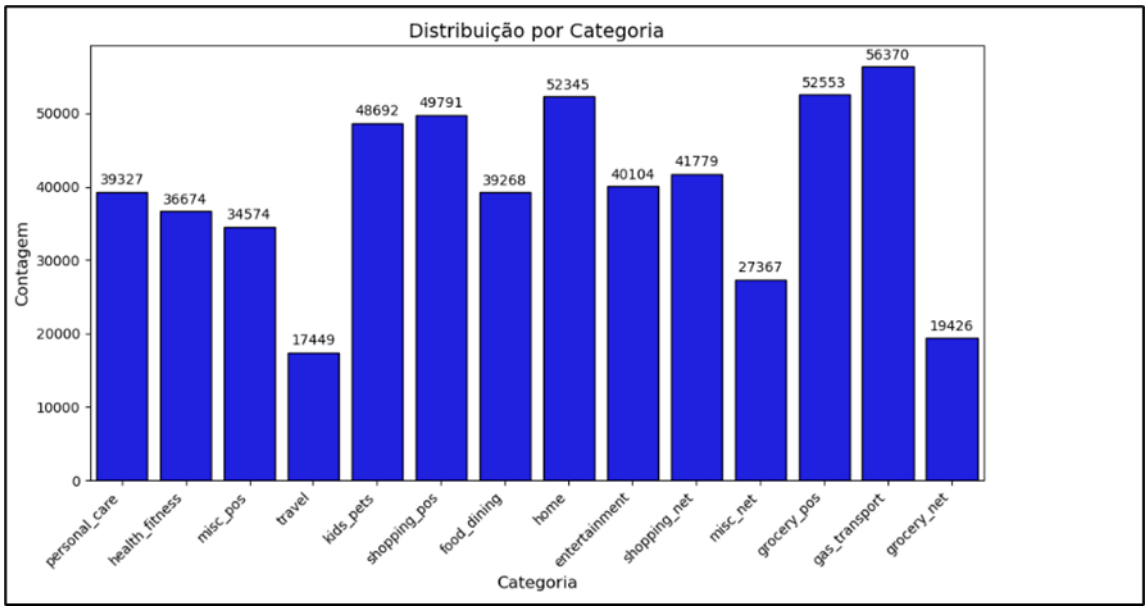


Figura 20: Distribuição por categoria

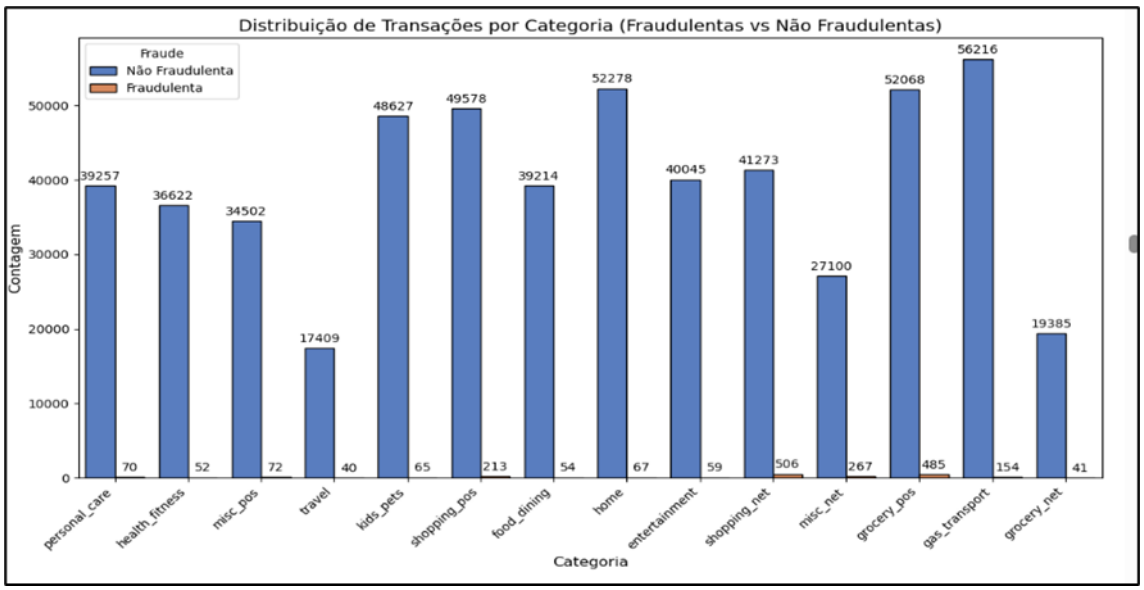


Figura 21: Distribuição de transações por categoria

A figura 20 mostra que as categorias "*gas_transport*" (56.370 transações), "*grocery_pos*" (52.553 transações) e "*home*" (52.345 transações) são as mais movimentadas, indicando que estas representam os principais tipos de despesas. Por outro lado, categorias como "*travel*" (17.449 transações) e "*grocery_net*" (19.426 transações) têm menor volume, possivelmente devido à sua natureza mais específica.

A figura 21 gráfico compara transações fraudulentas e não fraudulentas em cada categoria, onde é possível visualizar que determinadas categorias podem ser mais vulneráveis a fraudes, mesmo que o volume total de transações não seja tão elevado.

A análise revela que categorias com maior volume de transações, como "*gas_transport*" e "*grocery_pos*", também registam fraudes, mas numa proporção relativamente baixa. No entanto, categorias como "*misc_net*" apresentam uma vulnerabilidade maior, apesar do menor volume total de transações. Estas informações são cruciais para direcionar estratégias de **monitorização mais rigorosas em categorias específicas**, reforçando a prevenção de fraudes de forma eficaz.

Análise variável “age”:

A análise da distribuição de transações por faixa etária é fundamental para compreender o comportamento financeiro em diversas idades. Esta abordagem permite identificar padrões de consumo e possíveis vulnerabilidades associadas a fraudes em cada faixa etária. Com estas informações, é possível ajustar estratégias de prevenção e monitorização, direcionando esforços para os grupos mais suscetíveis.

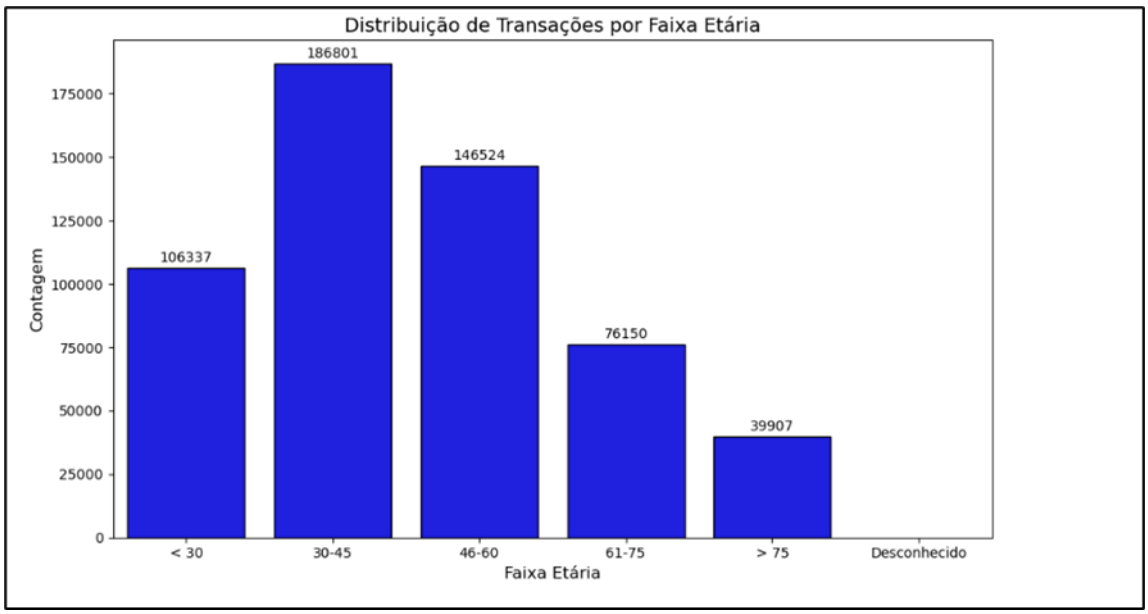


Figura 22: Distribuição de transações por faixa etária

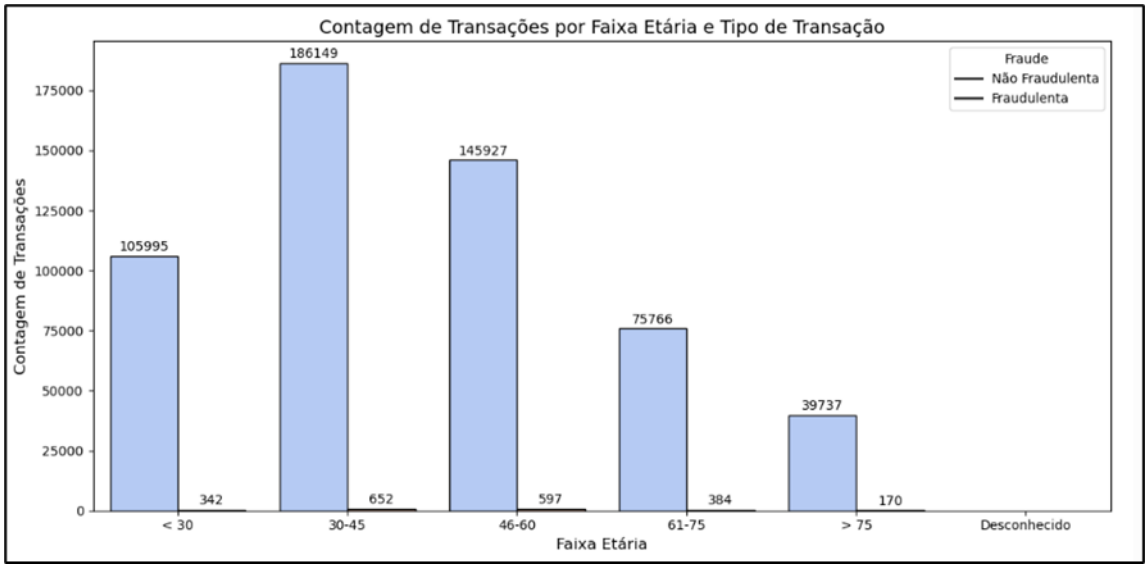


Figura 23: Contagem de transações por faixa etária e tipo de transação

Os gráficos apresentados mostram a distribuição de transações totais e fraudulentas por faixa etária. No primeiro gráfico, observa-se que a faixa etária entre 30 e 45 anos é a que apresenta o maior número de transações, com um total de 186.801, seguida pelas faixas 46 a 60 anos (146.524) e menos de 30 anos (106.337).

No segundo gráfico, que analisa a contagem de transações fraudulentas e não fraudulentas, observa-se que a proporção de fraudes é baixa em todas as faixas etárias. Ainda assim, o grupo 30 a 45 anos, que domina o volume de transações, também regista o maior número de fraudes, com 652 casos. Este padrão reflete-se em outras faixas etárias, onde a quantidade de fraudes está proporcionalmente alinhada ao número total de transações.

A análise mostra que as faixas etárias com maior atividade financeira, como 30 a 45 anos e 46 a 60 anos, são também as mais representadas em casos de fraude. Contudo, a proporção de fraudes é baixa em relação ao volume total de transações em todos os grupos etários. Estes resultados sugerem que o risco de fraude está mais relacionado ao volume de transações do que à faixa etária em si. Estratégias de prevenção devem, portanto, focar-se no comportamento transacional de cada grupo, garantindo uma monitorização equilibrada, sem preconceitos associados à idade.

Análise Variáveis Espaciais:

A análise geográfica das transações é essencial para identificar padrões regionais que possam influenciar tanto o volume total de transações como a ocorrência de fraudes. Ao compreender como estes comportamentos variam entre diferentes estados, é possível direcionar medidas de prevenção e monitorização para as regiões mais suscetíveis, otimizando os esforços de deteção de fraudes.

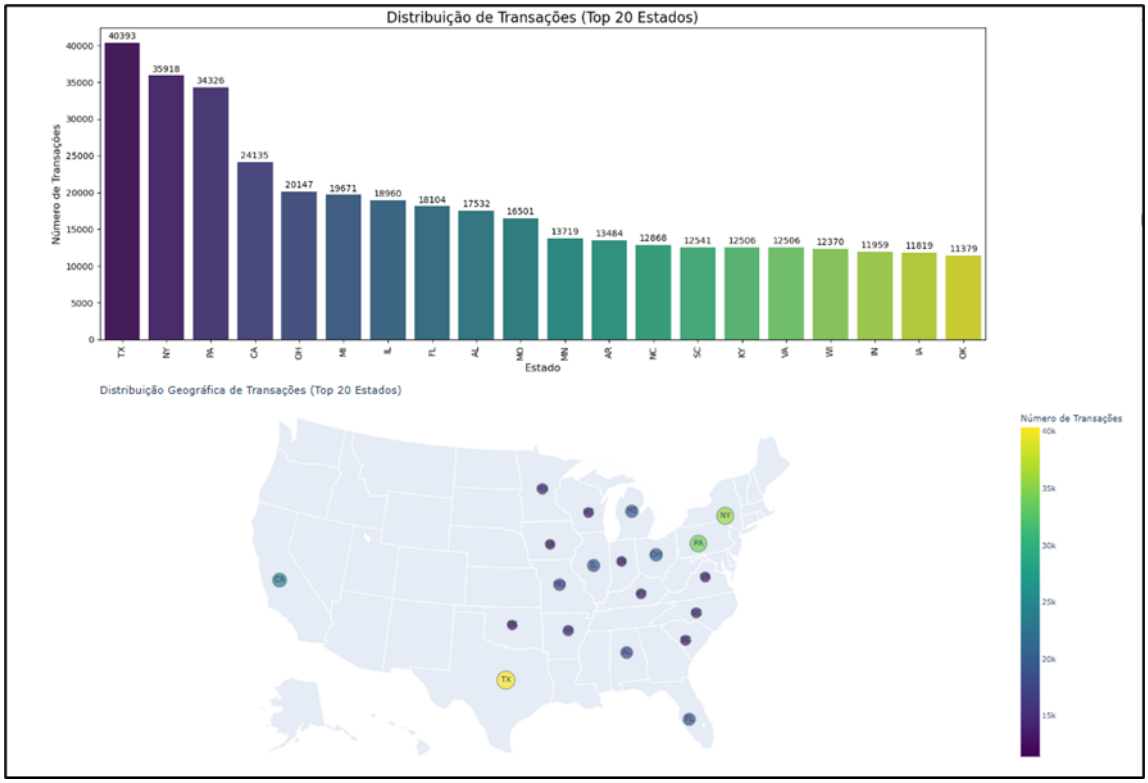


Figura 24: Distribuição de transações (top 20 estados)

O primeiro gráfico mostra que os estados com maior volume de transações são Texas (TX), com 40.393 transações, seguido por Nova Iorque (NY) com 35.918 e Pensilvânia (PA) com 34.326. Estes estados representam centros populacionais e comerciais significativos, o que justifica o elevado número de transações. Por outro lado, estados como Oklahoma (OK) e Iowa (IA) registam volumes significativamente mais baixos, com menos de 12.000 transações.

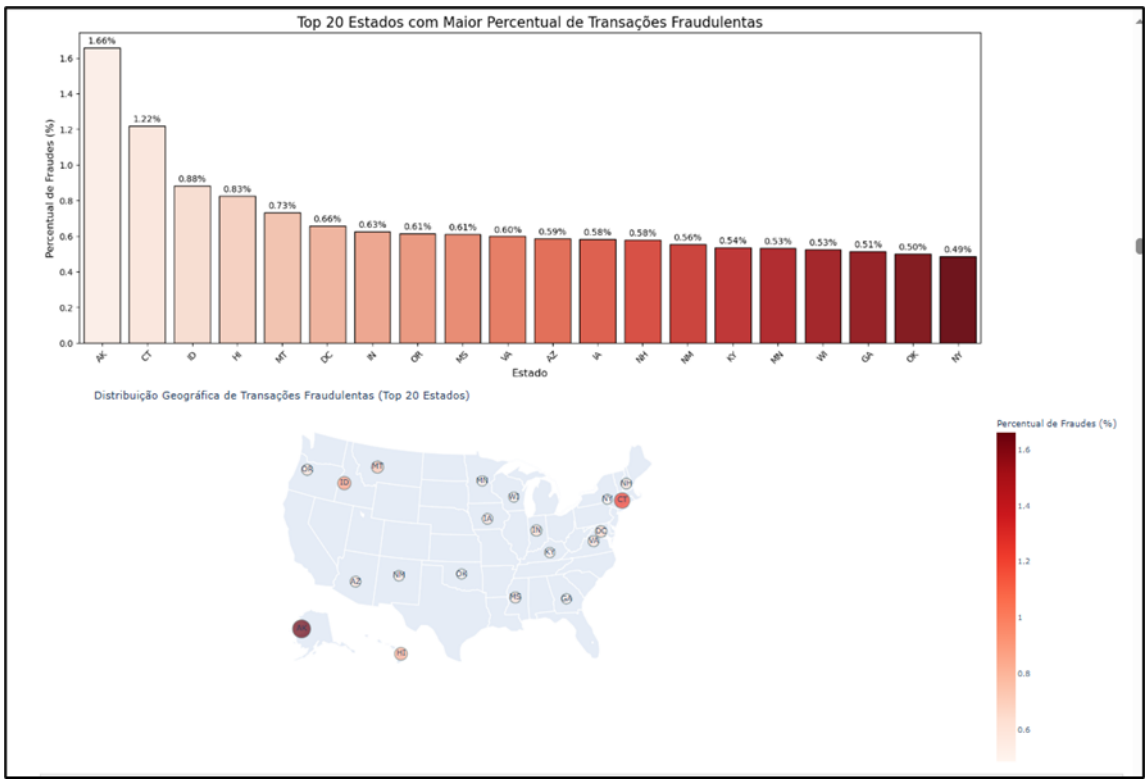


Figura 25: Top 20 estados com maior % de transações fraudulentas

No segundo gráfico, observa-se que os estados com maior percentagem de transações fraudulentas não são necessariamente os mesmos que lideram em volume total de transações. O Alasca (AK) apresenta o maior percentual de fraudes, com 1,66%, seguido por Connecticut (CT) com 1,22% e Dakota do Norte (ND) com 0,88%. Estes estados, apesar de não estarem entre os mais movimentados, destacam-se pela vulnerabilidade a fraudes.

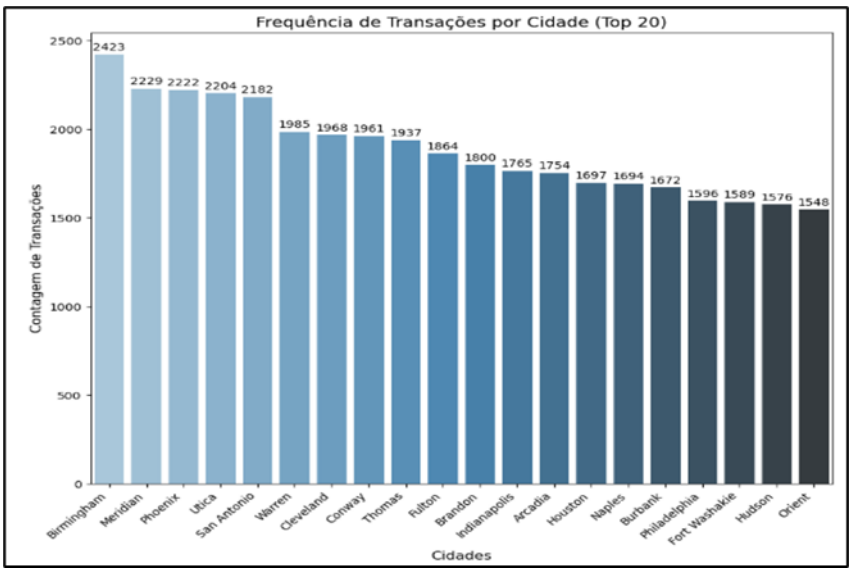


Figura 26: Frequência de transações por cidade (top 20)

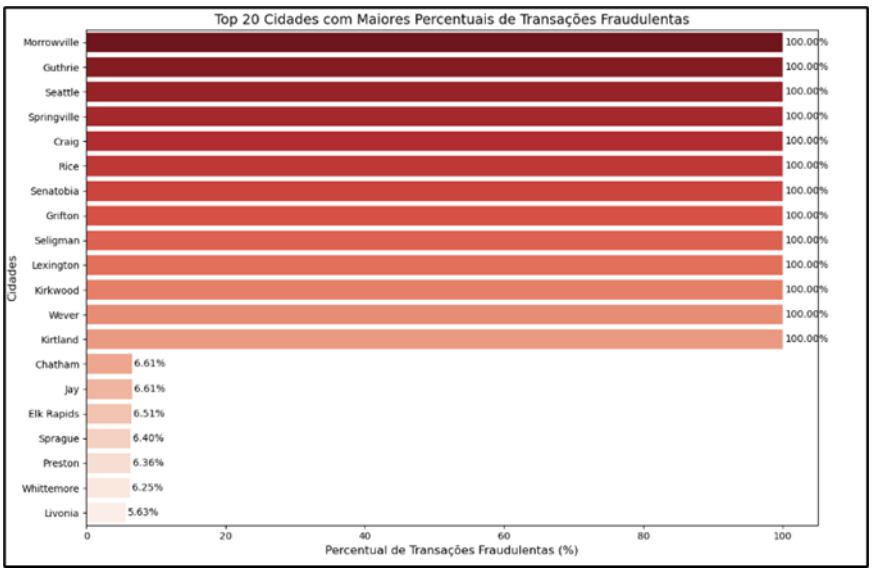


Figura 27: Top 20 cidades com maior % de transações fraudulentas

O Gráfico apresenta as 20 cidades com maior frequência de transações. A cidade de Birmingham destaca-se no topo com 2.423 transações, seguida de Meridian (2.229) e Phoenix (2.222). Estas cidades refletem centros urbanos ativos com grande volume de transações. Por outro lado, as cidades mais próximas do final do *ranking*, como Fort Wayne (1.576) e Orient (1.548), apresentam volumes menores, embora ainda estejam entre as 20 mais movimentadas.

A figura 27 foca-se nas cidades com maiores percentuais de fraudes. Cidades como Morrowville, Guthrie, Seattle e várias outras registam uma taxa de fraudes de 100%, o que indica que todas as transações registadas nestas cidades são fraudulentas. Este padrão pode ser causado por dados limitados, uma concentração de atividades fraudulentas em áreas menos movimentadas ou erros de registo.

Entre as cidades com percentagens inferiores, como Jay (6,61%) e Livonia (5,63%), observa-se um comportamento mais típico, com uma mistura de transações legítimas e fraudulentas. Estes valores refletem padrões de fraude mais equilibrados.

As cidades com maior volume de transações, como Birmingham e Meridian, não aparecem entre as que possuem maior percentagem de fraudes. Por outro lado, cidades com percentagens elevadas, como Morrowville e Guthrie, podem indicar áreas de alta concentração de fraude ou representatividade limitada de transações legítimas.

Análise Variável "*merchant*":

A análise dos comerciantes com maior volume de transações e maior percentagem de fraudes é fundamental para identificar os principais atores no sistema de transações financeiras e os potenciais focos de vulnerabilidade. Este tipo de avaliação ajuda a priorizar a monitorização em comerciantes específicos e a reforçar os sistemas de deteção de fraudes.

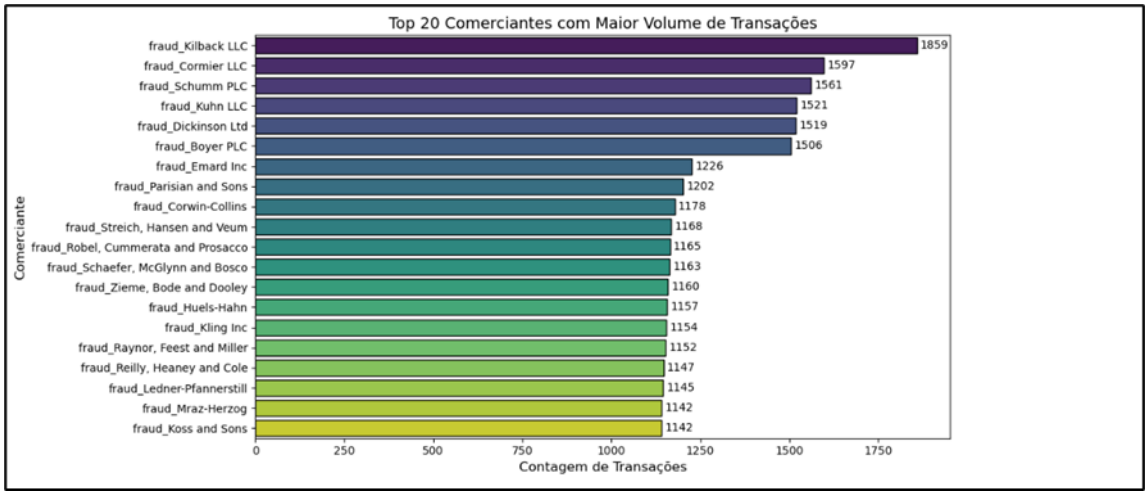


Figura 28: Top 20 comerciantes com maior volume de transações

O gráfico revela que o comerciante fraud_Kilback LLC lidera em volume de transações, com 1.859 transações, seguido por fraud_Cormier LLC (1.597) e fraud_Schumm PLC (1.561). Estes comerciantes destacam-se pelo elevado número de operações registadas, indicando forte atividade financeira. Os restantes comerciantes apresentam volumes entre 1.506 e 1.142 transações, mostrando uma concentração moderada de transações no topo da lista.

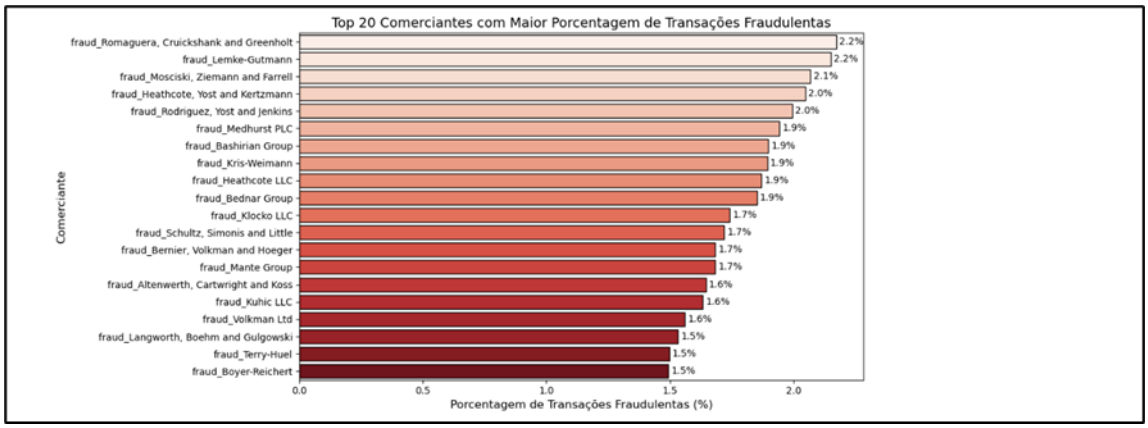


Figura 29: Top 20 comerciantes com maior % de transações fraudulentas

O segundo gráfico mostra os comerciantes com maior percentagem de fraudes. fraud_Romaguera, Cruickshank and Greenholt e fraud_Lemke-Gutmann lideram com uma taxa de 2,2% de transações fraudulentas, seguidos de perto por fraud_Mosciski, Ziemann and Farrell e fraud_Heathcote, Yost and Kertzmann,

ambos com 2,0%. Estes comerciantes apresentam uma maior suscetibilidade a fraudes, embora os valores percentuais sejam baixos no geral.

A análise revela que os comerciantes com maior volume de transações, como fraud_Kilback LLC, não coincidem necessariamente com os que têm maior percentagem de fraudes. Comerciantes como fraud_Romaguera, Cruickshank and Greenholt, apesar de terem uma taxa de fraude elevada, não estão entre os principais em volume de transações.

3. Preparação dos Dados

3.1. Seleção e Preparação dos Dados para Modelagem

A fase de Seleção e Preparação dos Dados é fundamental para garantir a qualidade e a relevância da informação utilizada no desenvolvimento do modelo preditivo. Nesta etapa, identificamos as variáveis mais importantes, eliminamos dados irrelevantes ou redundantes e transformamos as variáveis para facilitar o seu processamento pelos algoritmos de machine learning.

Adicionalmente, tratamos valores ausentes, realizamos codificações para lidar com variáveis categóricas e escalonamos variáveis contínuas, assegurando a consistência dos dados e reduzindo potenciais problemas de dimensionalidade.

Este processo é essencial para transformar os dados brutos em *inputs* adequados, maximizando o potencial do modelo para atingir o objetivo que, neste caso, é a deteção de fraudes financeiras.

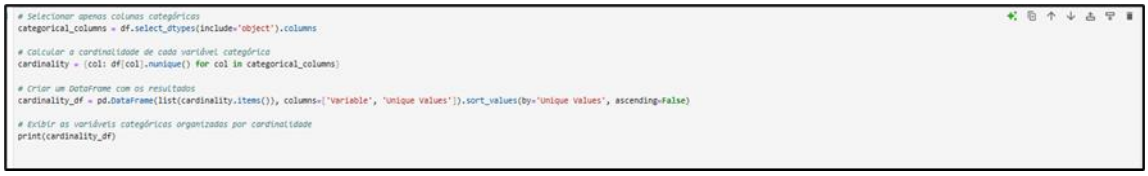
A análise inicial do conjunto de dados revelou um total de 555.719 registos e 23 colunas, que incluem variáveis numéricas, categóricas e temporais. Observou-se que o *dataset* não possui valores nulos em nenhuma variável, o que elimina a necessidade de tratamento inicial de valores ausentes.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 555719 entries, 0 to 555718
Data columns (total 23 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   cc_num             555719 non-null  float64
 1   merchant           555719 non-null  object
 2   category           555719 non-null  object
 3   amt                555719 non-null  float64
 4   gender             555719 non-null  object
 5   street             555719 non-null  object
 6   city               555719 non-null  object
 7   state              555719 non-null  object
 8   zip                555719 non-null  int64
 9   lat                555719 non-null  float64
 10  long               555719 non-null  float64
 11  city_pop           555719 non-null  int64
 12  job                555719 non-null  object
 13  trans_num          555719 non-null  object
 14  unix_time          555719 non-null  int64
 15  merch_lat          555719 non-null  float64
 16  merch_long         555719 non-null  float64
 17  is_fraud           555719 non-null  int64
 18  trans_hour         555719 non-null  int32
 19  trans_day_of_week  555719 non-null  object
 20  trans_year_month   555719 non-null  period[M]
 21  age                555719 non-null  int64
 22  age_category       555719 non-null  object
dtypes: float64(6), int32(1), int64(5), object(10), period[M](1)
memory usage: 95.4+ MB
```

Figura 30: Visualização valores nulos

No entanto, verificamos que o *dataset* continha uma diversidade de tipos de dados (como *object*, *float64, int32*, *int64* e *period[M]*), o que indica a necessidade de transformações específicas, permitindo a sua utilização adequada em análises e modelos preditivos.

Assim, e de forma a aprofundar a compreensão das variáveis categóricas, calculou-se a cardinalidade de cada uma delas, permitindo identificar aquelas com maior diversidade de valores e que requerem estratégias especificas de codificação.



```
# Selecionar apenas colunas categóricas
categorical_columns = df.select_dtypes(include='object').columns

# Calcular a cardinalidade de cada variável categórica
cardinality = {col: df[col].nunique() for col in categorical_columns}

# Criar um DataFrame com os resultados
cardinality_df = pd.DataFrame(list(cardinality.items()), columns=['Variable', 'Unique Values']).sort_values(by='Unique Values', ascending=False)

# Exibir as variáveis categóricas organizadas por cardinalidade
print(cardinality_df)
```

Figura 31: Cálculo cardinalidade

	Variable	Unique Values
7	trans_num	555719
3	street	924
4	city	849
0	merchant	693
6	job	478
5	state	50
1	category	14
8	trans_day_of_week	7
9	age_category	5
2	gender	2

Figura 32: Visualização valores únicos

Após o cálculo da cardinalidade das variáveis categóricas presentes no *dataset*, destacam-se as seguintes observações:

- A variável *trans_num* apresenta uma cardinalidade extremamente elevada (555.719 valores únicos), indicando que é um identificador exclusivo de transações. Por ser única para cada registo, será considerada irrelevante para a análise preditiva, mas decidimos manter a mesma por ser um indicador que pode permitir a instituição bancária a identificar a transação fraudulenta de uma forma muito mais ágil.

- Variáveis como *trans_num*, *street*, *city*, *merchant*, *job*, *state* e *category* possuem alta cardinalidade, exigindo estratégias de codificação adequadas, como *Target Encoding*.

- Variáveis como *gender*, *trans_day_of_week*, *age_category* e *trans_year_month* apresentam baixa cardinalidade, tornando-se candidatas a codificação direta, como *One-Hot Encoding*.

Esta análise é essencial para definir a estratégia de pré-processamento, otimizando a utilização das variáveis categóricas no modelo de deteção de fraudes.

Posto isto, avançamos para as transformações / *feature encoding*:

```python
# Criar uma cópia do dataframe original para o novo dataframe transformado
df1 = df.copy()

# 1. Target Encoding para variáveis de alta cardinalidade
high_cardinality_vars = ['trans_num', 'street', 'city', 'merchant', 'job', 'state', 'category']
for var in high_cardinality_vars:
    # Calcular a média da variável alvo (is_fraud) no dataframe original
    target_mean = df.groupby(var)['is_fraud'].mean()
    # Aplicar o target encoding no novo dataframe
    df1[var + '_target_enc'] = df1[var].map(target_mean)
    # Remover a variável original do novo dataframe
    df1.drop(var, axis=1, inplace=True)

# 2. One-Hot Encoding para variáveis de baixa cardinalidade
low_cardinality_vars = ['gender', 'trans_day_of_week', 'age_category', 'trans_year_month']

# Aplicar o One-Hot Encoding diretamente no novo dataframe
df1 = pd.get_dummies(df1, columns=low_cardinality_vars, drop_first=True)

# Verificar o novo dataframe transformado
print("Dataframe original (df):")
print(df.info())  # Verifica que o dataframe original permanece intacto
print("\nDataframe transformado (df1):")
print(df1.info())  # Verifica o dataframe transformado
```

Figura 33: Processo de Encoding

```
Dataframe transformado (df1):
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 555719 entries, 0 to 555718
Data columns (total 36 columns):
 #   Column                        Non-Null Count   Dtype
---  ------                        --------------   -----
 0   cc_num                        555719 non-null  float64
 1   amt                           555719 non-null  float64
 2   zip                           555719 non-null  int64
 3   lat                           555719 non-null  float64
 4   long                          555719 non-null  float64
 5   city_pop                      555719 non-null  int64
 6   unix_time                     555719 non-null  int64
 7   merch_lat                     555719 non-null  float64
 8   merch_long                    555719 non-null  float64
 9   is_fraud                      555719 non-null  int64
 10  trans_hour                    555719 non-null  int32
 11  age                           555719 non-null  int64
 12  trans_num_target_enc          555719 non-null  float64
 13  street_target_enc             555719 non-null  float64
 14  city_target_enc               555719 non-null  float64
 15  merchant_target_enc           555719 non-null  float64
 16  job_target_enc                555719 non-null  float64
 17  state_target_enc              555719 non-null  float64
 18  category_target_enc           555719 non-null  float64
 19  gender_M                      555719 non-null  bool
 20  trans_day_of_week_Monday      555719 non-null  bool
 21  trans_day_of_week_Saturday    555719 non-null  bool
 22  trans_day_of_week_Sunday      555719 non-null  bool
 23  trans_day_of_week_Thursday    555719 non-null  bool
 24  trans_day_of_week_Tuesday     555719 non-null  bool
 25  trans_day_of_week_Wednesday   555719 non-null  bool
 26  age_category_46-60            555719 non-null  bool
 27  age_category_61-75            555719 non-null  bool
 28  age_category_< 30             555719 non-null  bool
 29  age_category_> 75             555719 non-null  bool
 30  trans_year_month_2020-07      555719 non-null  bool
 31  trans_year_month_2020-08      555719 non-null  bool
 32  trans_year_month_2020-09      555719 non-null  bool
 33  trans_year_month_2020-10      555719 non-null  bool
 34  trans_year_month_2020-11      555719 non-null  bool
 35  trans_year_month_2020-12      555719 non-null  bool
dtypes: bool(17), float64(13), int32(1), int64(5)
memory usage: 87.4 MB
None
```

Figura 34: Visualização dataframe transformado

De referir que após este processo decidimos fazer *drop* da variável categórica *age*, uma vez que temos as idades divididas por faixa etária através de *binning*.

De seguida, criamos a matriz de correlação das variáveis acima e de seguida, verificamos quais as variáveis altamente correlacionadas entre si, uma vez que

este é um processo importante para evitar redundâncias e multicolinearidade, que podem comprometer a *performance* do modelo.

```
# Parte 1: Matriz de correlação
df_corr = df1.corr()

# Plotando o heatmap da matriz de correlação
plt.figure(figsize=(15, 15))
sns.heatmap(df_corr, annot=False, cmap='coolwarm', square=True, cbar=True)
plt.title("Matriz de Correlação do Dataset")
plt.show()
```

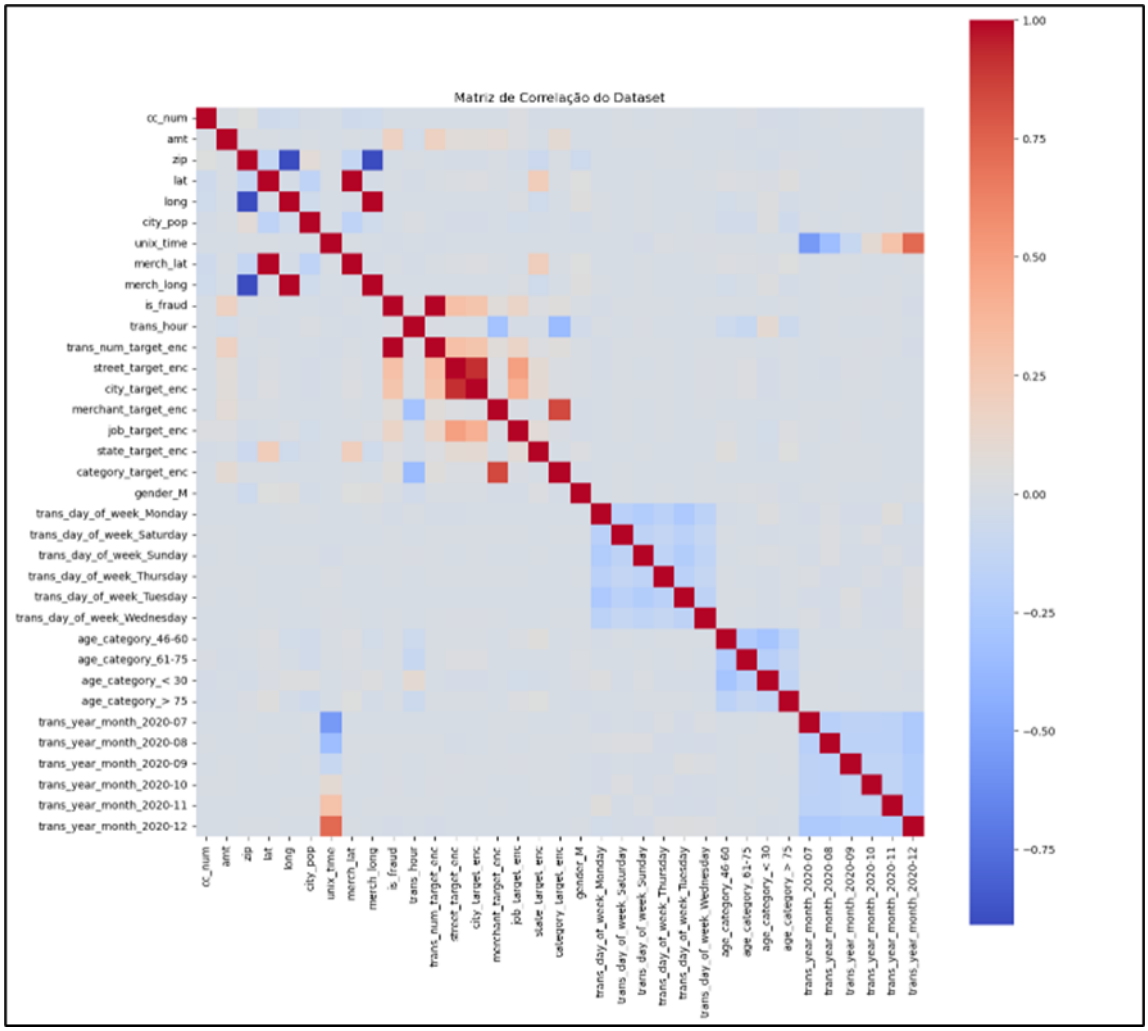


Figura 35: Matriz de correlação

```
# Identificar colunas altamente correlacionadas acima de 85%
corr_features = correlation(df1, 0.85)
print("Colunas altamente correlacionadas acima de 85%:")
print(corr_features)
```

```
Colunas altamente correlacionadas acima de 85%:
{'trans_num_target_enc', 'city_target_enc', 'merch_lat', 'merch_long'}
```

Figura 36: Colunas altamente correlacionadas (85% +)

Verificamos que as variáveis "*trans_num_target_enc*", "*city_target_enc*", "merch_lat" e "merch_long", foram identificadas, pelo que decidimos verificar a correlação das mesmas com a variável alvo:

```
# Verificar a correlação com a variável target
correlation_with_target = df1[['trans_num_target_enc', 'merch_long', 'city_target_enc', 'is_fraud']].corr()['is_fraud']
print(correlation_with_target)
```

```
trans_num_target_enc    1.000000
merch_long             -0.001060
city_target_enc         0.281274
is_fraud                1.000000
Name: is_fraud, dtype: float64
```

```
# Remover a variável menos correlacionada com o alvo ('merch_long')
df1.drop(columns=['merch_long'], inplace=True)
```

Figura 37: Verificar correlação com a variável target

Do *output* acima, podemos identificar que a variável "*trans_num_target_enc*" é a que apresenta a maior correlação com a variável alvo. No entanto, a mesma será deixada de fora da análise preditiva, conforme mencionamos acima.

Com base nos resultados, decidimos remover a variável *merch_long* por apresentar a menor correlação com a variável alvo, contribuindo assim para a simplificação e eficiência do modelo.

Após a análise da matriz de correlação e a remoção de variáveis altamente correlacionadas, foi implementado um modelo de *Random Forest* para calcular a importância das variáveis.

O objetivo desta etapa é identificar quais variáveis têm maior impacto na deteção de fraudes, otimizando o conjunto de dados para a modelação preditiva. Durante a preparação das variáveis, a variável *trans_num_target_enc* foi removida, uma vez que representa um identificador único de transações e não fornece informações estatísticas úteis, além de poder causar *overfitting*.

Para o treino do modelo, foi utilizado um *Random Forest* com parâmetros ajustados (n_estimators=100 e max_depth=10) para avaliar a importância relativa de cada variável no conjunto de dados. Os resultados desta análise foram extraídos e organizados numa tabela, destacando as variáveis mais relevantes, e um gráfico de barras foi gerado para visualizar o impacto relativo de cada variável de forma clara e objetiva. O objetivo final desta etapa é reduzir o ruído nos dados, removendo variáveis irrelevantes ou de baixa importância, o que contribui para melhorar o desempenho do modelo e facilita a sua interpretação. As variáveis mais importantes identificadas nesta análise servirão como base para o treino do modelo final.

```python
from sklearn.ensemble import RandomForestClassifier
import pandas as pd
import matplotlib.pyplot as plt

# Preparação das variáveis (X e y)
X = df1.drop(columns=['is_fraud'])  # Remover a variável alvo
y = df1['is_fraud']  # Variável alvo

# Remover a variável 'trans_num_target_enc' (ou qualquer identificador único)
if 'trans_num_target_enc' in X.columns:
    X = X.drop(columns=['trans_num_target_enc'])

# Treinamento de um modelo de Random Forest para obter as importâncias
rf = RandomForestClassifier(random_state=42, n_estimators=100, max_depth=10)
rf.fit(X, y)

# Obter a importância das features
feature_importance = pd.DataFrame({
    "Feature": X.columns,
    "Importance": rf.feature_importances_
}).sort_values(by="Importance", ascending=False)

# Formatar as importâncias para melhorar a visualização
feature_importance["Importance"] = feature_importance["Importance"].apply(lambda x: round(x, 4))

# Exibir a tabela usando o método padrão
display(feature_importance)

# Plotar gráfico de barras horizontais das importâncias
plt.figure(figsize=(10, 12))  # Gráfico vertical para acomodar todas as variáveis
plt.barh(feature_importance["Feature"], feature_importance["Importance"], color="teal")
plt.title("Importância das Variáveis - Random Forest", fontsize=16)
plt.xlabel("Importância", fontsize=12)
plt.ylabel("Variáveis", fontsize=12)
plt.gca().invert_yaxis()  # Inverter a ordem para que as maiores importâncias fiquem no topo
plt.tight_layout()
plt.show()
```

Figura 38: Desenvolvimento de código

Do código acima, resultou o seguinte gráfico demonstrativo da importância das variáveis:

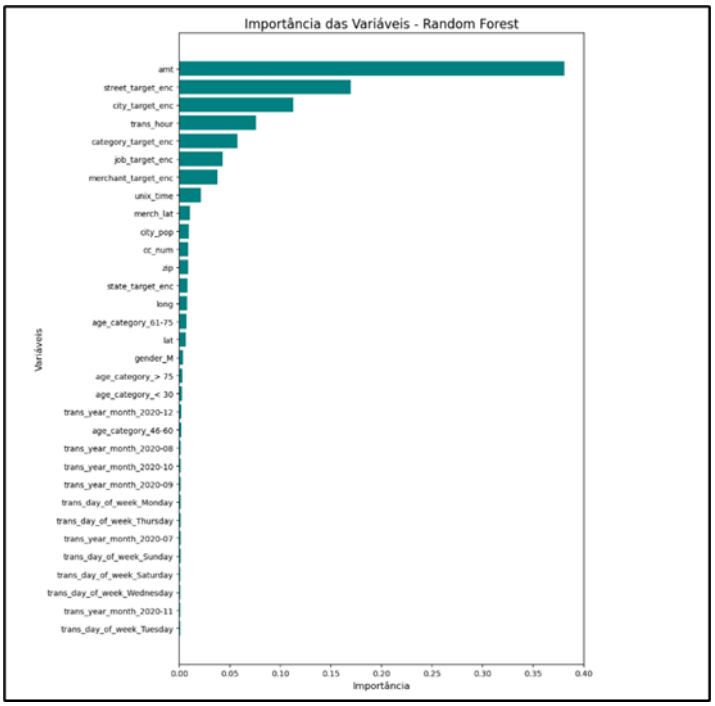


Figura 39: Importância das variáveis

A análise de importância das variáveis destaca *amt* como a *feature* mais relevante, seguida por *street_target_enc* e city_target_enc, que também demonstram uma contribuição significativa na deteção de fraudes. Com base nestes resultados, foi feita uma seleção estratégica das variáveis para inclusão no modelo final, priorizando aquelas com maior impacto preditivo, como:

- *amt, street_target_enc, city_target_enc, trans_hour, category_target_enc, job_target_enc, merchant_target_enc, unix_time, merch_lat, long e city_pop.*

Posteriormente, foi analisada a distribuição da variável alvo (*is_fraud*), observando-se o desbalanceamento natural entre as classes (fraude e não fraude), fenómeno comum em problemas de deteção de fraudes.

Para preparar os dados para a modelação, identificaram-se as variáveis contínuas que necessitavam de escalonamento, como *amt*, *trans_hour*, *unix_time*, entre outras, que foram normalizadas utilizando a técnica de padronização com o *StandardScaler*. Por fim, o conjunto de dados foi dividido em conjuntos de treino e teste, utilizando uma divisão estratificada, assegurando que a proporção da classe alvo (fraudes) fosse mantida em ambos os conjuntos.

Adicionalmente, as proporções foram verificadas e visualizadas, confirmando o balanceamento adequado.

```python
# 1. Verificar valores ausentes
print("\nVerificação de valores ausentes:")
missing_values = df1.isnull().sum()
print(missing_values[missing_values > 0])  # Exibe apenas as colunas com valores ausentes

# Preencher ou lidar com valores ausentes (se existirem)
df1.fillna(0, inplace=True)  # Aqui usamos 0 como padrão (ajustar conforme necessário)

# 2. Verificar redundâncias e identificadores únicos
print("\nVerificação de colunas redundantes ou identificadores únicos:")
print(df1.columns)

# Confirmar que 'trans_num_target_enc' ou variáveis irrelevantes não estão presentes
if 'trans_num_target_enc' in df1.columns:
    df1.drop(columns=['trans_num_target_enc'], inplace=True)

# 3. Selecionar apenas as variáveis mais importantes com base na análise
selected_features = [
    'amt', 'street_target_enc', 'city_target_enc', 'trans_hour',
    'category_target_enc', 'job_target_enc', 'merchant_target_enc',
    'unix_time', 'merch_lat', 'long', 'city_pop'
]

# Filtrar as colunas no DataFrame
df1 = df1[selected_features + ['is_fraud']]

print("\nVariáveis selecionadas para o modelo:")
print(selected_features)

# 4. Verificar balanceamento da variável alvo
print("\nDistribuição inicial da variável alvo (is_fraud):")
fraud_distribution = df1['is_fraud'].value_counts()
print(fraud_distribution)

# Visualizar a distribuição
plt.figure(figsize=(6, 4))
sns.barplot(x=fraud_distribution.index, y=fraud_distribution.values, palette="viridis")
plt.title("Distribuição da Variável Alvo (Antes do Balanceamento)", fontsize=14)
plt.xlabel("Classes (0 = Não Fraude, 1 = Fraude)", fontsize=12)
plt.ylabel("Contagem", fontsize=12)
plt.tight_layout()
plt.show()

# 5. Preparar para escalonamento (Normalização ou Padronização)
# Identificar variáveis contínuas para escalonamento
continuous_features = ['amt', 'trans_hour', 'unix_time', 'city_pop', 'merch_lat', 'long']

# Exibir variáveis contínuas confirmadas
print("\nVariáveis contínuas para escalonamento:")
print(continuous_features)

# Divisão em X e y
X = df1.drop(columns=['is_fraud'])  # Selecionar todas as variáveis preditoras
y = df1['is_fraud']  # Variável alvo

# Divisão em treino e teste (stratified split para balancear as classes)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, stratify=y)

# Escalonamento
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = X_train.copy()
X_test_scaled = X_test.copy()
X_train_scaled[continuous_features] = scaler.fit_transform(X_train[continuous_features])
X_test_scaled[continuous_features] = scaler.transform(X_test[continuous_features])

# Exibir proporções das classes no treino e teste
print("\nProporção da variável alvo no conjunto de treino:")
print(y_train.value_counts(normalize=True))

print("\nProporção da variável alvo no conjunto de teste:")
print(y_test.value_counts(normalize=True))

# Tabela de proporções das classes
class_distribution = pd.DataFrame({
    "Dataset": ["Treino", "Teste"],
    "Não Fraude (0)": [y_train.value_counts()[0], y_test.value_counts()[0]],
    "Fraude (1)": [y_train.value_counts()[1], y_test.value_counts()[1]],
    "Proporção Fraude": [y_train.value_counts(normalize=True)[1], y_test.value_counts(normalize=True)[1]]
})
print("\nDistribuição das Classes nos Conjuntos de Treino e Teste:")
print(class_distribution)

# Visualização das proporções em um gráfico
plt.figure(figsize=(8, 5))
sns.barplot(data=class_distribution.melt(id_vars="Dataset", var_name="Classe", value_name="Contagem"),
            x="Dataset", y="Contagem", hue="Classe", palette="viridis")
plt.title("Distribuição das Classes nos Conjuntos de Treino e Teste", fontsize=14)
plt.xlabel("Dataset", fontsize=12)
plt.ylabel("Contagem", fontsize=12)
plt.tight_layout()
```

Figura 40: Desenvolvimento de código

Da aplicação do código anterior, forma gerados os seguintes gráficos:

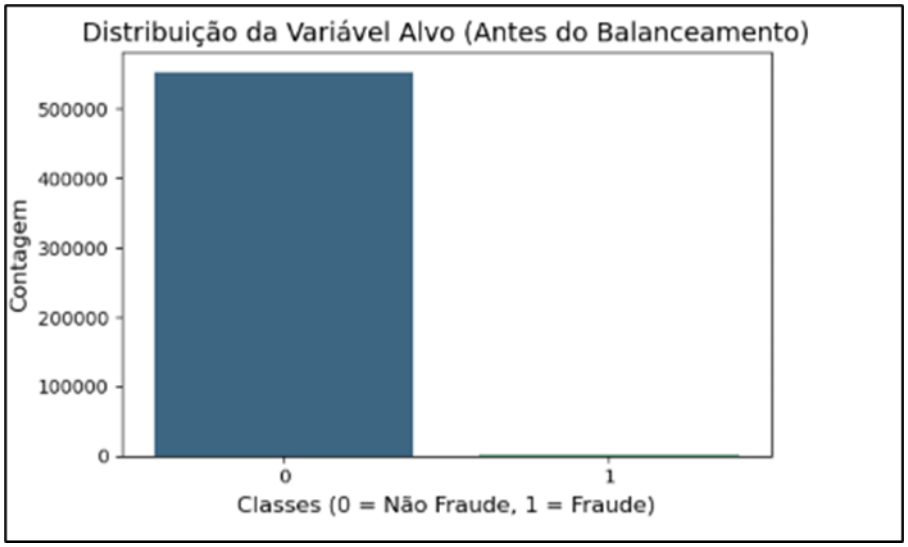


Figura 41: Distribuição da variável alvo

Este gráfico ilustra a distribuição inicial da variável alvo (*is_fraud*) no conjunto de dados. É evidente que existe um grande desbalanceamento entre as classes, com a classe 0 (Não Fraude) a dominar o conjunto de dados, enquanto a classe 1 (Fraude) apresenta uma frequência extremamente baixa.

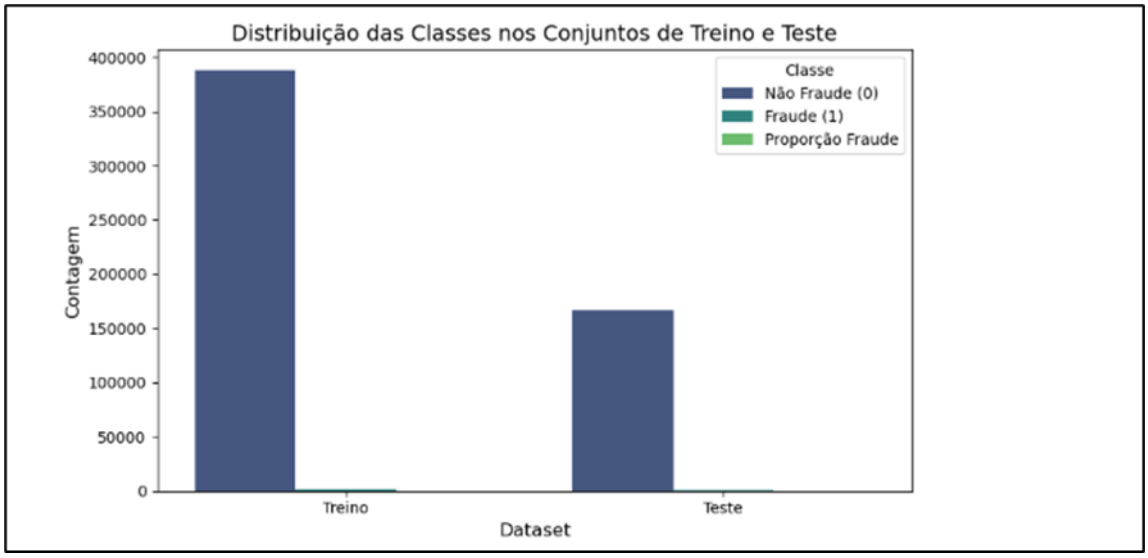


Figura 42: Distribuição das classes nos conjuntos de treino e teste

O segundo gráfico mostra a distribuição das classes nos conjuntos de treino e teste, após a divisão estratificada dos dados. Observa-se que a proporção entre Não Fraude (classe 0) e Fraude (classe 1) foi mantida nos dois conjuntos, o que é fundamental para garantir que ambos os conjuntos reflitam a realidade do dataset original.

Embora a proporção de fraudes continue baixa, a estratificação assegura que o modelo seja treinado e avaliado com base numa representação consistente das

classes, prevenindo enviesamentos que poderiam surgir de uma divisão aleatória.

Os gráficos destacam a necessidade de estratégias para lidar com o desbalanceamento da variável alvo, como *oversampling* ou penalizações durante a modelação, para melhorar a deteção de fraudes. Ao mesmo tempo, a divisão estratificada dos dados garante uma base sólida para o treino e avaliação do modelo, minimizando potenciais enviesamentos durante o processo de modelação.

4. Modelação

4.1. Modelos de Classificação Escolhidos

Para o desenvolvimento deste trabalho, foi necessária a implementação de diferentes modelos de *machine learning* com o objetivo de avaliar a sua capacidade de deteção de fraudes. A escolha dos modelos foi baseada na diversidade de abordagens que oferecem, desde métodos simples e interpretáveis, como a Regressão Logística, até modelos mais avançados, como *Gradient Boosting* e Redes Neurais. Esta variedade permite comparar diferentes estratégias, desde as que priorizam simplicidade e velocidade até aquelas que focam em precisão e capacidade de capturar relações complexas entre variáveis. Cada modelo selecionado apresenta características e vantagens específicas que o tornam relevante para esta análise.

1. Regressão Logística (*Logistic Regression*)

A regressão logística é um modelo estatístico simples e interpretável, usado para prever a probabilidade de um evento binário (como fraude ou não fraude). É escolhido pelo seu desempenho em problemas lineares e pela facilidade de interpretação dos coeficientes das variáveis.

Vantagens:

- Rápido e eficiente em *datasets* grandes.
- Fácil de interpretar e implementar.
- Bom desempenho em problemas lineares.

2. Árvore de Decisão (*Decision Tree*)

A árvore de decisão é um modelo não linear que utiliza uma estrutura hierárquica para dividir os dados com base em condições específicas. É escolhido pela sua simplicidade e capacidade de lidar com dados não lineares.

Vantagens:

- Fácil de interpretar e visualizar.
- Capaz de lidar com dados categóricos e contínuos.
- Não requer escalonamento de variáveis.

3. *Random Forest*

O *Random Forest* combina múltiplas árvores de decisão para melhorar a precisão e evitar *overfitting*. É um dos modelos mais robustos para deteção de fraudes devido à sua capacidade de generalização.

Vantagens:

- Resistente a *overfitting*.
- Excelente desempenho em *datasets* complexos.
- Lida bem com dados desbalanceados.

4. *Gradient Boosting*

Gradient Boosting é um modelo de *ensemble* que combina múltiplos modelos fracos (como árvores de decisão) para criar um modelo forte, otimizando o erro iterativamente. É escolhido pela sua elevada precisão em tarefas preditivas.

Vantagens:

- Elevada precisão.
- Robusto contra *overfitting* com ajustes adequados.
- Bom para problemas complexos e não lineares.

5. *XGBoost*

O *XGBoost* é uma variante otimizada do *Gradient Boosting* que apresenta melhorias em termos de velocidade e eficiência. É amplamente utilizado para tarefas de classificação devido ao seu excelente desempenho.

Vantagens:

- Rápido e eficiente.
- Excelente precisão em *datasets* grandes.
- Inclui regularização para evitar *overfitting*.

6. *Naive Bayes*

O *Naive Bayes* é um classificador probabilístico baseado no teorema de *Bayes*, assumindo independência entre as variáveis. É escolhido pela sua simplicidade e eficiência em problemas onde as hipóteses de independência são razoáveis.

Vantagens:

- Rápido e eficiente em *datasets* grandes.
- Bom desempenho com *datasets* extensos.
- Simples de implementar e interpretar.

7. Rede Neural MLP (*MLP Neural Network*)

A Rede Neural *Perceptron* Multicamadas (MLP) é um modelo mais complexo que pode capturar relações não lineares entre as variáveis. É escolhido pela sua capacidade de generalização em problemas complexos.

Vantagens:

- Lida bem com dados não lineares.
- Capaz de capturar relações complexas entre variáveis.
- Adaptável a diferentes tipos de dados.

4.2. Técnicas de Balanceamento

Para lidar com o desafio do desbalanceamento, foram aplicadas diferentes técnicas de balanceamento que ajustam a distribuição das classes, garantindo que o modelo tenha uma visão mais equilibrada das duas categorias. Desde o uso do *dataset* original, passando por técnicas como *Undersampling* e *Oversampling*, até métodos mais sofisticados como SMOTE e ADASYN, o objetivo é melhorar a capacidade do modelo de identificar transações fraudulentas, sem comprometer a generalização dos resultados.

1. Técnica Original sem Balanceamento

Nesta abordagem, o modelo é treinado utilizando o *dataset* original, sem realizar qualquer alteração na distribuição das classes. Serve como um *baseline* para comparar os resultados das outras técnicas.

Vantagens:

- Representa os dados reais sem manipulação.
- Permite avaliar o impacto direto do desbalanceamento no modelo.

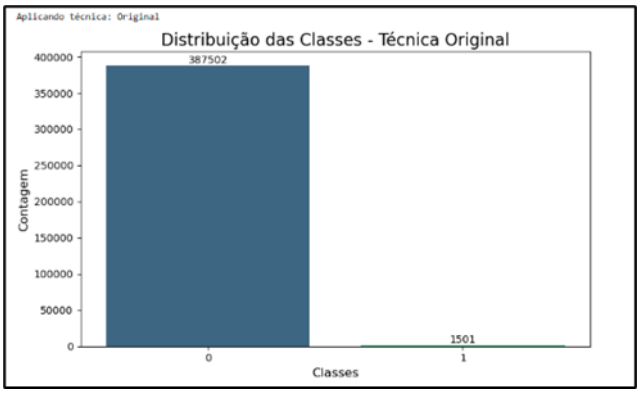


Figura 43: Distribuição das classes - Técnica original

2. *Undersampling*

O *undersampling* reduz a quantidade de dados da classe maioritária para igualar a da classe minoritária, criando um dataset balanceado. É útil quando o volume de dados não é crítico para a performance.

Vantagens:

- Simples de implementar.
- Reduz o tempo de treino devido à menor quantidade de dados.

Desvantagem:

- Pode perder informações relevantes da classe majoritária.

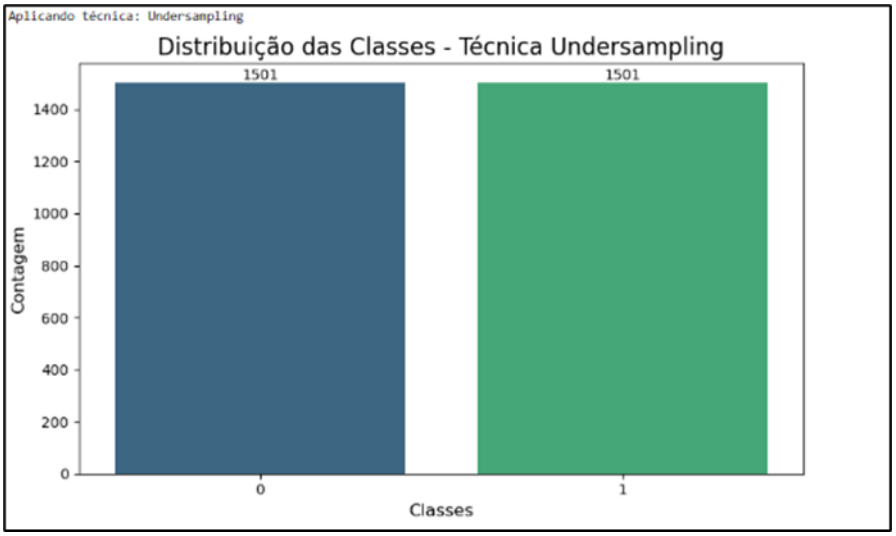


Figura 44: Distribuição das classes - Técnica Undersampling

3. *Oversampling*

O *oversampling* aumenta a quantidade de dados da classe minoritária ao replicar exemplos existentes ou criar novos exemplos. Ajuda a balancear o *dataset* sem reduzir a quantidade de dados.

Vantagens:

- Mantém todas as informações da classe maioritária.
- Melhora a capacidade do modelo de generalizar para a classe minoritária.

Desvantagens:

- Pode levar a *overfitting*, especialmente se os exemplos da classe minoritária forem simplesmente replicados.
- Aumenta o tempo de processamento devido ao aumento do tamanho do dataset.

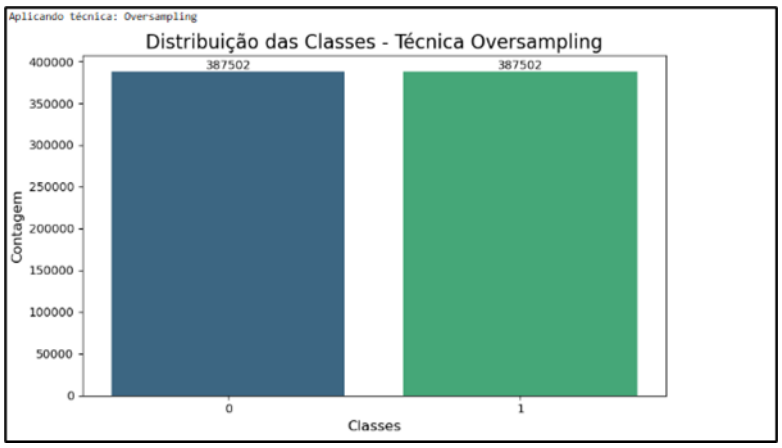


Figura 45: Distribuição das classes - Técnica Oversampling

4. SMOTE (*Synthetic Minority Oversampling Technique*)

O SMOTE cria novos exemplos sintéticos para a classe minoritária, em vez de simplesmente replicar os exemplos existentes. É amplamente utilizado para lidar com dados desbalanceados.

Vantagens:

- Gera exemplos mais variados para a classe minoritária.
- Reduz o risco de *overfitting* comparado ao *oversampling* tradicional.

Desvantagens:

- Pode introduzir ruído nos dados, pois os exemplos sintéticos são interpolados e podem não representar a realidade.
- Não considera a distribuição real dos dados, podendo criar exemplos menos representativos em certas regiões do espaço de decisão.

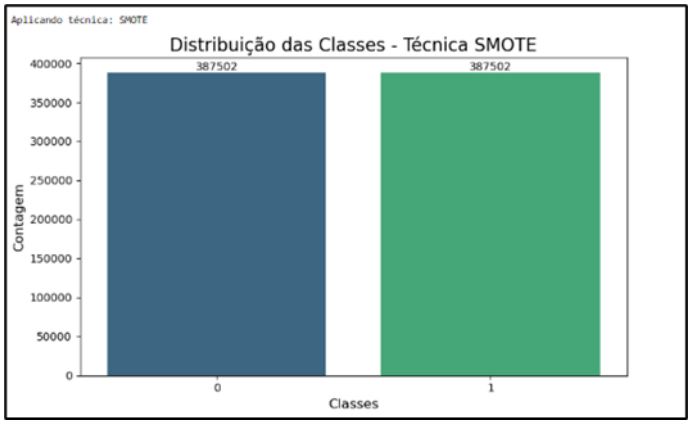


Figura 46: Distribuição das Classes - Técnica SMOTE

5. ADASYN (*Adaptive Synthetic Sampling*)

O ADASYN é uma extensão do SMOTE que gera exemplos sintéticos de forma adaptativa, concentrando-se nas regiões mais difíceis de classificar. É utilizado para melhorar a generalização do modelo.

Vantagens:

- Foca nas áreas mais desafiadoras do espaço de decisão.
- Reduz o desbalanceamento de forma mais inteligente e adaptativa.

Desvantagens:

- Tal como o SMOTE, pode introduzir ruído ao gerar exemplos sintéticos em regiões menos representativas.
- Exige maior processamento e ajuste para garantir que os exemplos sintéticos são úteis e não prejudicam a qualidade do modelo.

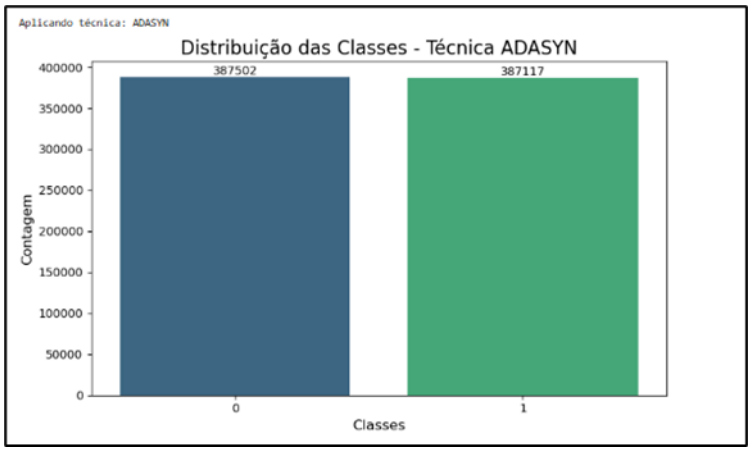


Figura 47: Distribuição das classes - Técnica ADASYN

5. Avaliação

A deteção de fraudes em transações com cartões de crédito é uma tarefa crítica e desafiadora, dada a raridade dos eventos fraudulentos em comparação com o volume total de transações. O objetivo principal desta etapa é identificar métodos eficientes que combinem algoritmos de *machine learning* e técnicas de manipulação de dados para maximizar a eficácia na identificação de fraudes. A configuração inicial desta análise foca-se na escolha de modelos preditivos capazes de lidar com *datasets* desbalanceados e complexos, garantindo uma base sólida para as etapas subsequentes.

A escolha é deliberada para incluir uma gama diversa de algoritmos – desde modelos lineares interpretáveis até técnicas avançadas como redes neurais e algoritmos baseados em árvores. Esta abordagem permite explorar diferentes formas de capturar padrões e interações nas variáveis, maximizando a probabilidade de detetar fraudes de forma eficiente.

```python
# Parte 1: Configuração Inicial e Modelos
import pandas as pd
import numpy as np
from sklearn.metrics import (roc_auc_score, confusion_matrix, accuracy_score, precision_score, recall_score, f1_score,
                             ConfusionMatrixDisplay, roc_curve, auc)
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
from xgboost import XGBClassifier
from imblearn.over_sampling import RandomOverSampler, SMOTE, ADASYN
from imblearn.under_sampling import RandomUnderSampler
import matplotlib.pyplot as plt
import seaborn as sns

# Configuração inicial
np.random.seed(42)

# Modelos
models = {
    "Logistic Regression": LogisticRegression(random_state=42),
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "Random Forest": RandomForestClassifier(random_state=42),
    "Gradient Boosting": GradientBoostingClassifier(random_state=42),
    "XGBoost": XGBClassifier(random_state=42, use_label_encoder=False, eval_metric="logloss"),
    "Naive Bayes": GaussianNB(),
    "MLP Neural Network (Balanced)": MLPClassifier(
        random_state=42,
        max_iter=200,  # Aumentar o número de iterações para maior estabilidade
        hidden_layer_sizes=(64, 32),  # 2 camadas escondidas com tamanhos equilibrados
        activation="relu",  # Função de ativação ReLU
        solver="adam"  # Otimizador Adam para bom desempenho
    )
}
```

Figura 48: Desenvolvimento de código

Nesta primeira etapa do código, são importadas todas as bibliotecas essenciais para o gerar os modelos:

- ***pandas e numpy*:** Para manipulação de dados e cálculos numéricos.
- ***matplotlib e seaborn*:** Para visualizações de gráficos, como a matriz de confusão e a curva ROC.
- ***sklearn*:** Biblioteca principal para *machine learning*, usada para definir modelos e calcular métricas de avaliação (*precisão*, *recall*, *F1-Score*, etc.).
- ***xgboost:*** Para implementação de modelos *XGBoost*, amplamente utilizados em problemas de classificação com dados desbalanceados.

- ***imblearn:*** Biblioteca específica para técnicas de balanceamento de classes, como *Oversampling*, *Undersampling*, SMOTE e ADASYN.

A configuração da semente aleatória garante a reprodutibilidade dos resultados. Em tarefas como a divisão do dataset em treino e teste, ou na aplicação de técnicas de balanceamento, a ordem dos dados e os valores aleatórios podem variar entre execuções. Fixar a semente (42, neste caso) assegura que os resultados sejam consistentes em diferentes execuções do código.

Nesta etapa, são definidos os modelos de *machine learning* que serão testados.

Modelos Selecionados:

1. ***Logistic Regression***
2. ***Decision Tree***
3. ***Random Forest***
4. ***Gradient Boosting***
5. ***XGBoost***
6. ***Naive Bayes***
7. ***MLP Neural Network (Balanced)***

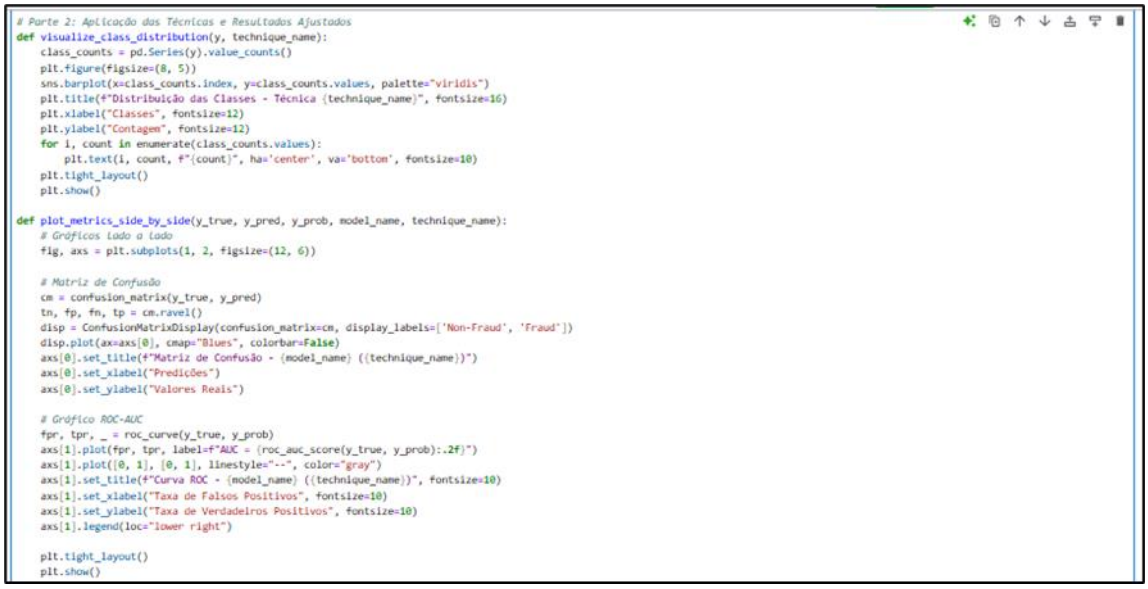
- Rede neural *Perceptron* Multicamadas (MLP) configurada com duas camadas escondidas de tamanhos equilibrados (64 e 32 neurónios).
- Utiliza a função de ativação ReLU e o optimizador Adam, que são eficientes para problemas complexos e com muitos dados.
- Vantagem: Capaz de capturar padrões não-lineares de alta complexidade, sendo flexível para diferentes tipos de dados.
- Limitação: Requer mais tempo e recursos computacionais para treino, comparado com modelos baseados em árvores.

Nesta segunda etapa (figura 49), foi configurado o ambiente necessário para aplicar o código de *machine learning*. A definição de múltiplos modelos assegura uma análise abrangente, permitindo explorar abordagens diferentes para o problema de deteção de fraudes

A parte central do código aplica as técnicas de balanceamento e os modelos definidos, avaliando o seu desempenho detalhadamente. O desbalanceamento das classes, comum em *datasets* de deteção de fraudes, é tratado com ajustes que garantem que os modelos identifiquem padrões de fraude sem favorecer a classe maioritária.

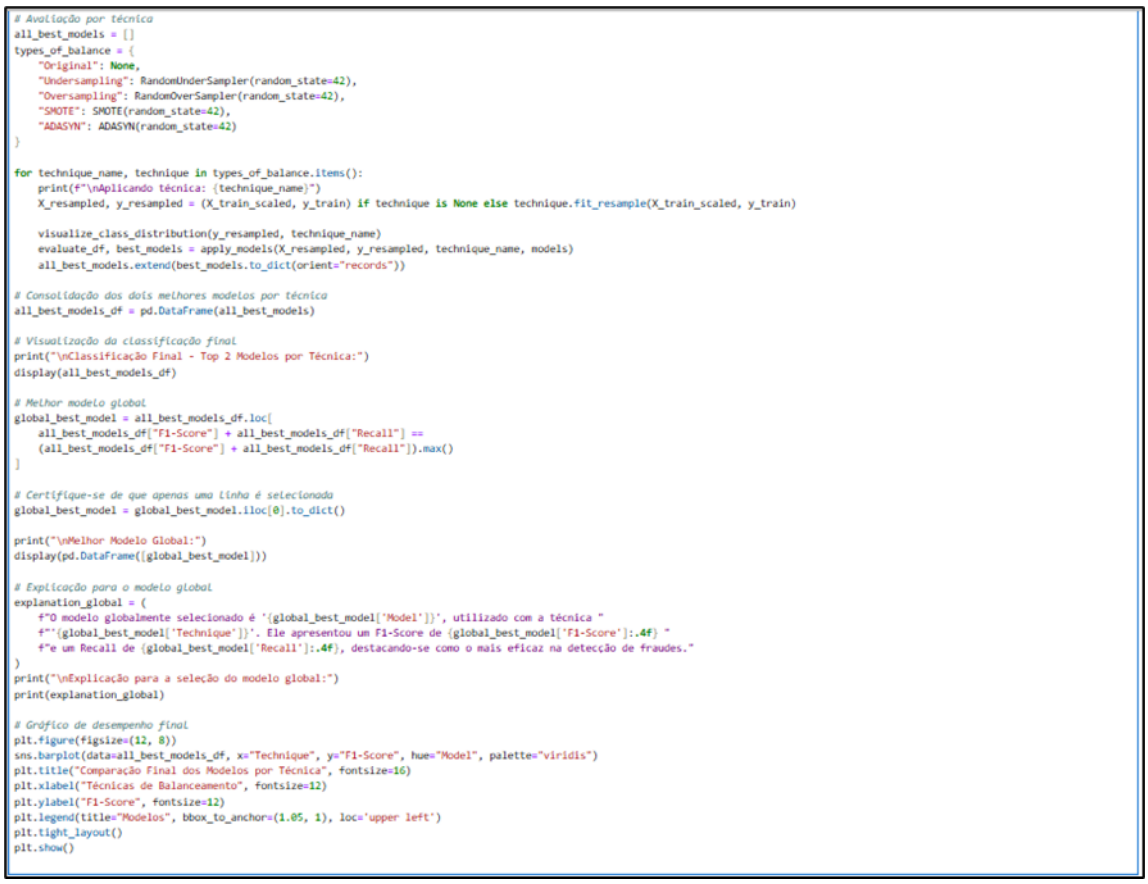
São utilizadas métricas como *F1-Score*, *Recall*, AUC e Precisão, com foco em maximizar a identificação de fraudes (*Recall*) enquanto se minimizam falsos positivos (FP) e falsos negativos (FN), devido ao impacto desses erros.

A precisão (*Precision*) é particularmente desafiante, pois a elevada proporção de transações legítimas pode levar a falsos positivos excessivos, gerando custos operacionais elevados. Assim, as métricas devem ser analisadas em conjunto para equilibrar a identificação de fraudes com a redução de erros.



```python
# Parte 2: Aplicação das Técnicas e Resultados Ajustados
def visualize_class_distribution(y, technique_name):
    class_counts = pd.Series(y).value_counts()
    plt.figure(figsize=(8, 5))
    sns.barplot(x=class_counts.index, y=class_counts.values, palette="viridis")
    plt.title(f"Distribuição das Classes - Técnica {technique_name}", fontsize=16)
    plt.xlabel("Classes", fontsize=12)
    plt.ylabel("Contagem", fontsize=12)
    for i, count in enumerate(class_counts.values):
        plt.text(i, count, f"{count}", ha='center', va='bottom', fontsize=10)
    plt.tight_layout()
    plt.show()

def plot_metrics_side_by_side(y_true, y_pred, y_prob, model_name, technique_name):
    # Gráficos lado a lado
    fig, axs = plt.subplots(1, 2, figsize=(12, 6))

    # Matriz de Confusão
    cm = confusion_matrix(y_true, y_pred)
    tn, fp, fn, tp = cm.ravel()
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Non-Fraud', 'Fraud'])
    disp.plot(ax=axs[0], cmap="Blues", colorbar=False)
    axs[0].set_title(f"Matriz de Confusão - {model_name} ({technique_name})")
    axs[0].set_xlabel("Predições")
    axs[0].set_ylabel("Valores Reais")

    # Gráfico ROC-AUC
    fpr, tpr, _ = roc_curve(y_true, y_prob)
    axs[1].plot(fpr, tpr, label=f"AUC = {roc_auc_score(y_true, y_prob):.2f}")
    axs[1].plot([0, 1], [0, 1], linestyle="--", color="gray")
    axs[1].set_title(f"Curva ROC - {model_name} ({technique_name})", fontsize=10)
    axs[1].set_xlabel("Taxa de Falsos Positivos", fontsize=10)
    axs[1].set_ylabel("Taxa de Verdadeiros Positivos", fontsize=10)
    axs[1].legend(loc="lower right")

    plt.tight_layout()
    plt.show()
```

```python
def explain_selection(best_model, technique_name):
    explanation = (
        f"O modelo selecionado para a técnica '{technique_name}' é '{best_model['Model']}'. "
        f"Este modelo apresentou um F1-Score de {best_model['F1-Score']:.4f} e um Recall de "
        f"{best_model['Recall']:.4f}, que são métricas essenciais para detecção de fraudes, pois "
        f"focam tanto no equilíbrio do desempenho quanto na identificação de fraudes com alta precisão."
    )
    print("\nExplicação para a seleção do modelo:")
    print(explanation)

def apply_models(X_resampled, y_resampled, technique_name, models_dict):
    evaluate_df = pd.DataFrame(columns=[
        "Model", "Train Score", "Test Score", "Accuracy", "F1-Score", "Precision", "Recall",
        "True Positives", "False Positives", "True Negatives", "False Negatives",
        "Precision TP", "Precision TN", "AUC", "Technique"
    ])

    for model_name, model in models_dict.items():
        model.fit(X_resampled, y_resampled)

        y_pred = model.predict(X_test_scaled)
        y_pred_prob = model.predict_proba(X_test_scaled)[:, 1] if hasattr(model, "predict_proba") else None

        acc_score = accuracy_score(y_test, y_pred)
        f_score = f1_score(y_test, y_pred, average='weighted')
        precision = precision_score(y_test, y_pred)
        recall = recall_score(y_test, y_pred)

        cm = confusion_matrix(y_test, y_pred)
        tn, fp, fn, tp = cm.ravel()
        precision_tp = tp / (tp + fp) if (tp + fp) > 0 else 0
        precision_tn = tn / (tn + fn) if (tn + fn) > 0 else 0
        auc = roc_auc_score(y_test, y_pred_prob) if y_pred_prob is not None else None

        evaluate_df = pd.concat([evaluate_df, pd.DataFrame([{
            "Model": model_name,
            "Train Score": model.score(X_resampled, y_resampled),
            "Test Score": model.score(X_test_scaled, y_test),
            "Accuracy": acc_score,
            "F1-Score": f_score,
            "Precision": precision,
            "Recall": recall,
            "True Positives": tp,
            "False Positives": fp,
            "True Negatives": tn,
            "False Negatives": fn,
            "Precision TP": precision_tp,
            "Precision TN": precision_tn,
            "AUC": auc,
            "Technique": technique_name
        }])], ignore_index=True)

        # Gráficos lado a lado
        plot_metrics_side_by_side(y_test, y_pred, y_pred_prob, model_name, technique_name)

    print(f"\nClassificação dos Modelos - Técnica: {technique_name}")
    display(evaluate_df)

    best_models = evaluate_df.nlargest(2, ["F1-Score", "Recall"])
    explain_selection(best_models.iloc[0].to_dict(), technique_name)

    return evaluate_df, best_models
```



```python
# Avaliação por técnica
all_best_models = []
types_of_balance = {
    "Original": None,
    "Undersampling": RandomUnderSampler(random_state=42),
    "Oversampling": RandomOverSampler(random_state=42),
    "SMOTE": SMOTE(random_state=42),
    "ADASYN": ADASYN(random_state=42)
}

for technique_name, technique in types_of_balance.items():
    print(f"\nAplicando técnica: {technique_name}")
    X_resampled, y_resampled = (X_train_scaled, y_train) if technique is None else technique.fit_resample(X_train_scaled, y_train)

    visualize_class_distribution(y_resampled, technique_name)
    evaluate_df, best_models = apply_models(X_resampled, y_resampled, technique_name, models)
    all_best_models.extend(best_models.to_dict(orient="records"))

# Consolidação dos dois melhores modelos por técnica
all_best_models_df = pd.DataFrame(all_best_models)

# Visualização da classificação final
print("\nClassificação Final - Top 2 Modelos por Técnica:")
display(all_best_models_df)

# Melhor modelo global
global_best_model = all_best_models_df.loc[
    all_best_models_df["F1-Score"] + all_best_models_df["Recall"] ==
    (all_best_models_df["F1-Score"] + all_best_models_df["Recall"]).max()
]

# Certifique-se de que apenas uma linha é selecionada
global_best_model = global_best_model.iloc[0].to_dict()

print("\nMelhor Modelo Global:")
display(pd.DataFrame([global_best_model]))

# Explicação para o modelo global
explanation_global = (
    f"O modelo globalmente selecionado é '{global_best_model['Model']}', utilizado com a técnica "
    f"'{global_best_model['Technique']}'. Ele apresentou um F1-Score de {global_best_model['F1-Score']:.4f} "
    f"e um Recall de {global_best_model['Recall']:.4f}, destacando-se como o mais eficaz na detecção de fraudes."
)
print("\nExplicação para a seleção do modelo global:")
print(explanation_global)

# Gráfico de desempenho final
plt.figure(figsize=(12, 8))
sns.barplot(data=all_best_models_df, x="Technique", y="F1-Score", hue="Model", palette="viridis")
plt.title("Comparação Final dos Modelos por Técnica", fontsize=16)
plt.xlabel("Técnicas de Balanceamento", fontsize=12)
plt.ylabel("F1-Score", fontsize=12)
plt.legend(title="Modelos", bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```

Figura 49: Desenvolvimento de código

Passos Principais e Objetivos

1. Visualização da Distribuição de Classes

A função *visualize_class_distribution* apresenta graficamente a distribuição das classes (fraude e não fraude) após a aplicação de cada técnica de balanceamento.

2. Avaliação do Desempenho com Métricas Gráficas

A função *plot_metrics_side_by_side* gera gráficos para avaliar o desempenho de cada modelo:

- Matriz de Confusão: Mostra a relação entre previsões corretas e incorretas (Verdadeiros Positivos, Falsos Positivos, etc.).
- Curva ROC-AUC: Mede a capacidade do modelo de distinguir entre fraudes e transações legítimas.

3. Explicação da Seleção de Modelos

A função *explain_selection* documenta a escolha do melhor modelo para cada técnica de balanceamento com base nas métricas calculadas.

4. Aplicação dos Modelos às Técnicas de Balanceamento

A função *apply_models* executa o treino e a avaliação de cada modelo para os dados processados com as diferentes técnicas de balanceamento:

Técnicas aplicadas:

- **Original:** Sem balanceamento.
- **Undersampling:** Reduz a classe maioritária para igualar a classe minoritária.
- **Oversampling:** Duplica ou sintetiza exemplos da classe minoritária para equilibrar as classes.
- **SMOTE:** Gera exemplos sintéticos da classe minoritária através de interpolação.
- **ADASYN:** Gera exemplos sintéticos, focando nas áreas mais difíceis de classificar.

5. Consolidação e Comparação dos Resultados

Os resultados de todas as combinações de técnicas e modelos são consolidados em um *DataFrame (all_best_models_df).* Posteriormente, o código:

- Identifica os dois melhores modelos por técnica com base no *Recall* e no *F1-Score*.

- Seleciona o melhor modelo global, considerando o equilíbrio entre *Recall*, Precisão e outros critérios.

6. Visualização e Análise Final

O gráfico final compara o *F1-Score* dos modelos para cada técnica de balanceamento.

Este código completo visa resolver o desafio do desbalanceamento em problemas de deteção de fraudes, testando uma ampla gama de modelos e técnicas de balanceamento. Assim, garante uma análise robusta, medindo o desempenho em métricas essenciais para o problema, como *Recall* (prioritário para identificar fraudes), *F1-Score* (equilíbrio entre *Recall* e Precisão) e AUC (capacidade de separação entre classes).

O foco está em identificar fraudes com alta precisão, mantendo baixos os falsos positivos, o que é crítico para evitar alarmes desnecessários, custos operacionais elevados e insatisfação dos clientes.

5.1. Avaliação dos Resultados

A seleção do modelo ideal para deteção de fraudes em cartões de crédito é uma etapa crucial, dado que o objetivo principal é identificar transações fraudulentas com elevada precisão e minimizar erros, principalmente os falsos positivos e os falsos negativos.

Nesta secção, apresentamos o melhor modelo selecionado, a técnica de balanceamento aplicada, e analisamos detalhadamente as métricas de desempenho. É importante sublinhar que a escolha do modelo se baseou numa avaliação ponderada das métricas-chave, considerando o contexto da deteção de fraudes, onde as consequências de cada tipo de erro (falso positivo ou falso negativo) podem ter impactos significativos.

1. *F1-Score* (0.998434):

O *F1-Score* é uma métrica que combina precisão (*precision*) e *recall* (sensibilidade), representando um equilíbrio entre a capacidade de prever corretamente fraudes e a fiabilidade das predições positivas. O elevado *F1-Score* do modelo demonstra um desempenho consistente e eficaz, essencial para um cenário sensível como a deteção de fraudes.

2. *Precision* (0.733840):

A precisão mede a proporção de predições classificadas como fraudes que realmente são fraudulentas. Neste caso, o valor de 0.733840 reflete que, embora o modelo tenha um bom desempenho geral, há uma proporção de falsos positivos que precisa de ser considerada. Contudo, o número de falsos positivos (210) é aceitável face à complexidade da tarefa.

3. *Recall* (0.899068):

O *recall* indica a proporção de fraudes identificadas corretamente pelo modelo. O elevado valor de 0.899068 mostra que o modelo é eficaz na deteção da maioria das fraudes, o que é crucial para minimizar perdas financeiras e aumentar a segurança.

4. *Accuracy* (0.998350):

A precisão global, ou *accuracy*, indica que quase todas as predições feitas pelo modelo são corretas. No entanto, como o *dataset* é desbalanceado, esta métrica sozinha não é suficiente para avaliar o desempenho real. Métricas como o *F1-Score* e o *recall* oferecem uma visão mais detalhada.

5. *True Positives* (579) e *False Negatives* (65):

O modelo identificou corretamente 579 fraudes e deixou de identificar apenas 65, o que demonstra um desempenho robusto na deteção de fraudes. A redução do número de falsos negativos é especialmente relevante, pois estas são fraudes

que podem passar despercebidas, resultando em perdas financeiras significativas.

6. *False Positives* (210) e *True Negatives* (165862):

O modelo classificou incorretamente 210 transações legítimas como fraudulentas, o que pode gerar inconvenientes para os clientes e custos operacionais adicionais. Contudo, o elevado número de verdadeiros negativos (165862) demonstra que o modelo tem uma boa capacidade de identificar corretamente transações legítimas.

7. AUC (0.998820):

A métrica AUC (*Area Under the Curve*) reflete a capacidade geral do modelo em distinguir entre classes (fraudes e não fraudes). O valor próximo de 1 (0.998820) é indicativo de um modelo com excelente capacidade discriminativa.

Conclusões:

O modelo *XGBoost* combinado com a técnica ADASYN destacou-se como o melhor modelo global devido aos seguintes fatores:

- Excelente *F1-Score* (0.998434), indicando um equilíbrio notável entre precisão e *recall*.
- Elevado *Recall* (0.899068), que assegura que a maioria das fraudes foi identificada, reduzindo significativamente o risco de perdas financeiras.
- Apesar de apresentar 210 falsos positivos, o modelo manteve uma elevada proporção de verdadeiros positivos e verdadeiros negativos, demonstrando um desempenho robusto em todas as frentes.

Esta configuração é uma escolha sólida para a deteção de fraudes, uma vez que prioriza a identificação de fraudes (*recall*) sem comprometer significativamente a precisão. Contudo, em contextos operacionais, a proporção de falsos positivos pode ser ajustada de acordo com a sensibilidade ao cliente e os custos associados a investigações manuais.

O modelo *XGBoost* com ADASYN é, assim, uma solução eficiente e equilibrada para deteção de fraudes em cartões de crédito, combinando alta performance com uma abordagem prática para os desafios do mundo real.

5.2. Estudo de Impacto Financeiro na Aplicação do Modelo Selecionado

Após a seleção do modelo *XGBoost* combinado com a técnica de balanceamento ADASYN como o mais eficiente para a deteção de fraudes, decidimos criar um estudo meramente académico para simular a aplicação prática deste modelo no contexto de um banco. Este estudo tem como objetivo avaliar, em termos financeiros, os impactos da implementação do modelo na redução de prejuízos, nos custos administrativos e na capacidade de identificar e bloquear transações fraudulentas.

O estudo considera diferentes métricas financeiras e operacionais para analisar o desempenho do modelo. Em particular, avaliam-se os seguintes componentes:

1. O prejuízo evitado, que corresponde ao montante das transações fraudulentas corretamente detetadas.

2. O prejuízo causado por fraudes não detetadas, ou seja, o impacto financeiro das transações fraudulentas que escaparam ao modelo.

3. Os custos administrativos associados a falsos positivos, que representam o esforço operacional necessário para investigar transações legítimas erroneamente classificadas como suspeitas.

4. Os custos administrativos decorrentes de fraudes não detetadas, que incluem o impacto adicional relacionado com a gestão destas transações.

Ao calcular estes componentes, conseguimos não apenas determinar o impacto financeiro líquido da implementação do modelo, mas também identificar potenciais áreas de melhoria e avaliar o custo-benefício da sua aplicação em larga escala. Segue-se uma análise detalhada de cada um dos fatores e dos seus resultados financeiros.

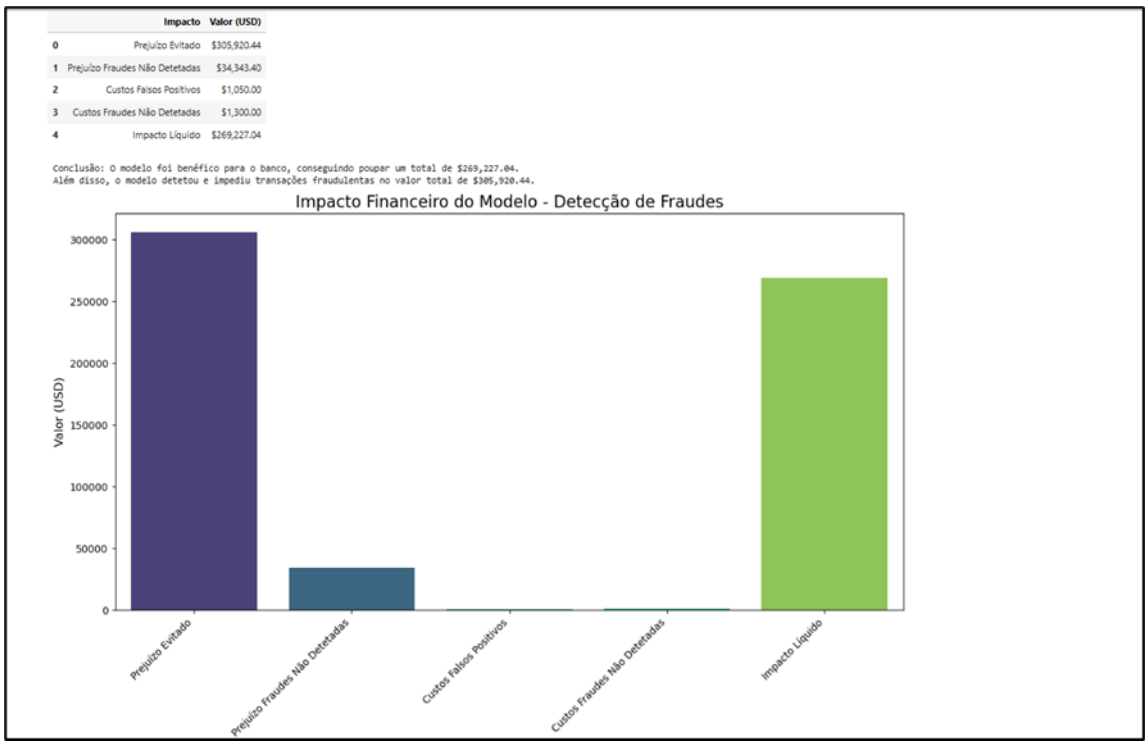


Figura 50: Impacto Financeiro do Modelo - Deteção de fraudes

Análise dos Valores

1. **Prejuízo Evitado: 305,920.44$**

- Este valor é calculado a partir das 579 transações fraudulentas corretamente identificadas (*True Positives*), multiplicadas pelo valor médio de uma transação fraudulenta (528.36$).
- Este montante reflete o benefício mais significativo do modelo, ao proteger diretamente os lucros do banco e mitigar perdas financeiras que seriam inevitáveis sem a aplicação do modelo.
- A elevada capacidade do modelo de detetar fraudes destaca a sua eficiência prática e a importância da sua utilização como ferramenta de proteção.

2. **Prejuízo com Fraudes Não Detetadas: 34,343.40$**

- Este valor considera as 65 fraudes não detetadas (False Negatives), multiplicadas pelo mesmo valor médio de transações fraudulentas ($528.36).
- Apesar de representar uma fração relativamente pequena do prejuízo evitado, estas fraudes não detetadas indicam que o modelo pode ser melhorado para aumentar a cobertura (*recall*) e capturar ainda mais transações fraudulentas.

3. **Custos Administrativos de Falsos Positivos: 1,050.00$**

- Este valor reflete o custo associado à investigação de 210 falsos positivos, com um custo administrativo estimado de 5$ por transação.
- Embora este custo seja relativamente baixo no contexto geral, pode ser otimizado para melhorar a experiência do cliente, reduzindo interrupções em transações legítimas e minimizando esforços desnecessários.

4. Custos Administrativos de Fraudes Não Detetadas: 1,300.00$

- Este valor considera um custo administrativo adicional de 20$ por cada fraude não detetada (65 *False Negatives*), relacionado com a gestão e resolução de transações fraudulentas não identificadas inicialmente.
- Este custo, embora pequeno, reforça a necessidade de aprimorar o *recall* do modelo para reduzir ao máximo as fraudes que escapam ao sistema.

Impacto Financeiro Total

Após a contabilização dos benefícios e custos, o impacto financeiro líquido do modelo foi calculado como 269,227.04$. Este valor positivo demonstra claramente que a aplicação do modelo gerou benefícios financeiros substanciais para o banco, superando amplamente os custos administrativos e os prejuízos das fraudes não detetadas.

Futuramente, esforços podem ser direcionados para:

1. Melhorar o *recall* do modelo, de forma a reduzir ainda mais o número de fraudes não detetadas.

2. Minimizar o número de falsos positivos, otimizando os custos administrativos e a satisfação do cliente.

6. Referências

Belfo, F. (2020). *Apresentação resumida e adaptada do modelo CRISP-DM*. Coimbra: ISCAC | Coimbra Business School.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). CRISP-DM 1.0: *Step-by-step data mining guide*. U.S.A.: SPSS, CRISP-DM Consortium.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

HR Portugal. (2022). *Millennials são as maiores vítimas de fraude financeira*. Disponível em https://hrportugal.sapo.pt.

IBM. (2022). *Estudo sobre impacto de fraudes financeiras*. Disponível em https://brasil.newsroom.ibm.com.

Phua, C., Lee, V., Smith, K., & Gayler, R. (2010). *A Comprehensive Survey of Data Mining-based Fraud Detection Research*. arXiv preprint arXiv:1009.6119.

Ribeiro, B. (2021). Fraude Financeira: Desafios e Soluções no Contexto Digital. Lisboa: Editora Minerva.

Thomas, E. J., & Rao, S. (2020). *Big Data Analytics for Banking and Fraud Detection*. Elsevier.

Trading Economics. (2024). *United States credit card accounts*. Disponível em https://tradingeconomics.com.

TransUnion. (2023). *Relatório anual de tendências de fraude digital omnichannel*. Disponível em https://www.transunion.com.br.