

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data

Yuri Ramos Felix de Souza

ANÁLISE DE RECUPERAÇÃO DE CRÉDITO DE VEÍCULOS

Belo Horizonte
2023

Yuri Ramos Felix de Souza

ANÁLISE DE RECUPERAÇÃO DE CRÉDITO DE VEÍCULOS

Trabalho de Conclusão de Curso apresentado
ao Curso de Especialização em Ciência de
Dados e Big Data como requisito parcial à
obtenção do título de especialista.

Belo Horizonte

2023

AGRADECIMENTOS

Minha gratidão a Deus, por ter me proporcionado a capacidade para sempre prosseguir em meus estudos.

Gratidão à minha família por sempre ter me apoiado e acompanhado em meu caminho.

Também agradeço à minha amiga Caroline Almuinha por ter me ajudado com sobre este trabalho.

Resumo

Este trabalho tem como objetivo coletar, tratar e analisar dados de uma série mensal de créditos a serem recuperados a fim de identificar potenciais pagamentos com base em conteúdo privado disponibilizado pelo banco Itaú Unibanco, utilizando a linguagem Python em sua versão 3.12.0 para os fins citados acima e para a construção de algoritmos de Machine Learning, junto com a plataforma Jupyter Notebook 6.5.4.

SUMÁRIO

1. Introdução	6
1.1. Contextualização	6
1.2. O problema proposto	7
1.3. Linguagem de Programação e Recursos Computacionais	9
2. Coleta de Dados	10
3. Processamento/Tratamento de Dados	11
4. Análise e Exploração dos Dados	19
5. Criação de Modelos de Machine Learning	24
6. Interpretação dos Resultados	35
7. Apresentação dos Resultados	38
8. Links	39
REFERÊNCIAS	40
APÊNDICE – Código do notebook utilizado no projeto	41

1. Introdução

1.1. Contextualização

Possuir um veículo próprio proporciona maior independência e flexibilidade de locomoção. O proprietário pode se deslocar para onde quiser a qualquer momento, sem depender de horários de transporte público, é mais conveniente para atividades diárias, com ir ao trabalho, levar as crianças à escola, fazer compras e outras tarefas cotidianas.

Em muitas regiões do Brasil, especialmente áreas rurais ou com menor infraestrutura de transporte público, um carro pode ser essencial para o acesso a locais remotos. Em áreas sujeitas a condições climáticas adversas, ter um carro oferece maior conforto e segurança para enfrentar chuvas fortes, enchentes ou outras situações climáticas desafiadoras e, em algumas profissões ou setores de negócios, ter um carro é uma necessidade para visitar clientes, fornecedores ou se deslocar para reuniões em locais diferentes.



Figura 1 – Proprietário dirigindo seu veículo.

Comprar um veículo próprio pode ser um grande desafio financeiro para as pessoas, onde muitas não possuem o poder aquisitivo necessário para usufruir deste produto e então devem buscar algumas estratégias que podem ajudar a superar as dificuldades financeiras ao comprar um carro, como: economizar, considerar opções usadas, financiamentos ou programas de descontos e incentivos.

O Itaú Unibanco é o maior banco privado brasileiro em valor de mercado e a marca mais valiosa do Brasil. Possui um leque de diversos serviços, incluindo financiamento de veículos. Este serviço possibilita as pessoas de possuírem seus próprios veículos com uma linha de crédito e que devem ser feitos os pagamentos das parcelas até a sua quitação.

1.2. O problema proposto

Temos hoje em nosso país diferentes tipos de financiamentos, e as opções podem variar de acordo com a instituição financeira, as condições de mercado e até mesmo a legislação vigente. Aqui estão alguns dos tipos comuns de financiamento de veículos:

- CDC (Crédito Direto ao Consumidor): Neste tipo de financiamento, o cliente toma um empréstimo para adquirir o veículo. O veículo fica em nome do comprador desde o início, mas o banco ou instituição financeira tem uma garantia até que o pagamento seja concluído.

- Consórcio: No consórcio, um grupo de pessoas se une para formar um fundo comum. Mensalmente, alguns membros são contemplados e recebem uma carta de crédito para a compra do veículo. Todos os participantes têm a chance de serem contemplados ao longo do tempo.

- Empréstimo Pessoal: Alguns consumidores optam por utilizar um empréstimo pessoal para financiar a compra de um veículo. Nesse caso, não há alienação fiduciária, mas o veículo é utilizado como garantia.

O banco Itaú Unibanco tem uma modalidade para o financiamento de veículos, que fornece o crédito necessário para seus clientes adquirirem seus

carros, com as formas de pagamento desses empréstimos acordado entre as partes. Mas ocorrem desses pagamentos não serem devidamente cumpridos, podendo levar a uma série de medidas tomadas por parte do credor para sanar este problema.

Quando ocorrem os atrasos nos pagamentos das parcelas, o credor (Itaú Unibanco) inicia seu processo interno de cobrança, realizando o primeiro contato dessa etapa. Mas o banco não possui interesse em se estender nessa negociação, terceirizando esse serviço de cobrança para outras empresas especializadas. O banco disponibiliza um conjunto de contratos e os envia para essas empresas para iniciarem um novo processo de cobrança, desde as tentativas de contatos com os clientes, passando por processos judiciais e, até, a recuperação dos veículos e leilão.

O banco envia um conjunto com milhares de dados para as recuperadoras de crédito, que devem efetuar a carga desses registros em suas bases de dados, fazer uma análise do que está entrando para a cobrança e avaliar as estratégias que serão utilizadas para ter uma melhor performance dessa arrecadação mensal. No final de cada mês, essas empresas devem realizar um comitê para demonstrar ao cliente o que foi recuperado e seus desempenhos.

Os dados disponibilizados pelo Itaú Unibanco são privados, então devem ser baixados diretamente de seu sistema, sem o uso de robôs. São dados de clientes, ou seja, sensíveis, e devem ser tomadas as medidas cabíveis para protegê-los. O arquivo .csv com estes dados serão disponibilizados em um link do google drive.

Serão analisados os saldos e períodos de atraso dos contratos, onde avaliaremos as potenciais recuperações. Os contratos que não forem tão favoráveis para essa recuperação deverão passar por um processo estratégico por parte da gerência e diretoria.

Todos os contratos são pertencentes à região do estado do Rio de Janeiro, visto que o banco distribui seus contratos com restrição geográfica para cada empresa de recuperação de crédito veicular.

Será analisado o período de um mês para a recuperação dos valores, onde será definido um padrão de análise.

1.3. Linguagem de Programação e Recursos Computacionais

Para coleta, exploração, manipulação, tratamento e análise dos dados foi utilizada a linguagem de programação Python, em sua versão 3.12.0, tendo seus códigos sido compilados no Jupyter Notebook, em sua versão 6.5.4.

O projeto conta com a base de dados descrita abaixo, devendo ser executada:

00_tcc_dadoscoletados.xlsx

A máquina utilizada para o projeto, ou seja, para a execução dos notebooks, conta com configuração de processador Intel Core i5, 16GB RAM, 250GB ROM e Windows 10 (Enterprise).

2. Coleta de Dados

Os dados disponíveis nesta fonte de dados (o site) não estão disponíveis para download direto. A empresa contratada deve realizar seu acesso por login e senha disponibilizadas pelo banco para poder realizar os downloads dos arquivos manualmente. A utilização de robôs não é permitida pelo banco, por motivos de segurança.

Os arquivos coletados contam com igual estrutura explicada na tabela abaixo:

Nome da coluna	Descrição	Tipo
CANAL	Grupo a que pertence o contrato a depender do atraso	object
GRUPO	Grupo a que pertence o contrato	object
FLAG_MAIORES_IMPACTOS	Impacto a depender do valor e atraso	object
SALDO_PDD_POTENCIAL	Última parcela devedora	object
SALDO_ESTOQUE_PDD	Saldo recuperado	object
NM_CLIENTE	Nome do cliente	object
REGIAO	Região do contrato	object
SALDO	Saldo total devedor do contrato	object
DT_PAGTO	Data do vencimento da parcela	object
DT_EXCLUSAO	Data da exclusão da parcela do sistema	object
TIPO_PAGTO	Definição do tipo de pagamento	object
NR_CONTRATO	Número do contrato	object
BASE	Base de extração do contrato	object
NR_DIAS_ATRASO	Total dos dias de atraso	object

Tabela 1 – Estrutura dos arquivos que compõem o conjunto de dados.

3. Processamento/Tratamento de Dados

Como os dados são privados, o arquivo .xlsx estará disponibilizado para download no link:

https://docs.google.com/spreadsheets/d/1c4gpm8MKSG6NXZgSZzDml2AmMaLPxOR5/edit?usp=drive_link&oid=108751598473419779391&rtpof=true&sd=true

Para iniciar o tratamento dos dados é lido o diretório TCC e é listado o arquivo existente nele (00_tcc_dadoscoletados.xlsx).

```
#-----#  
# SELECIONANDO ARQUIVO DE DADOS #  
#-----#  
  
#-----#  
# Varrendo diretório onde está localizado o arquivo de dados #  
#-----#  
  
# Definindo nome do diretório  
path = r"C:\TCC\00_tcc_dadoscoletados.xlsx"  
path  
  
df = pd.read_excel(path, engine='openpyxl')  
df
```

Figura 2 – Listando arquivos que foram baixados.

Inicialmente, o conjunto de dados conta com um total de 10272 registros e 13 colunas.

```
#-----#
# IMPORTANDO REGISTROS DO ARQUIVO DE DADOS LISTADOS #
#-----#

# Verificar informações de total de registros e colunas
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10272 entries, 0 to 10271
Data columns (total 13 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   CANAL                                10272 non-null  object
1   GRUPO                                10272 non-null  object
2   FLAG_MAIORES_IMPACTOS                10272 non-null  object
3   SALDO_PDD_POTENCIAL                  10272 non-null  float64
4   SALDO_ESTOQUE_PDD                    10272 non-null  float64
5   NM_CLIENTE                           10272 non-null  object
6   SALDO                                 10272 non-null  float64
7   DT_PAGTO                             5305 non-null   datetime64[ns]
8   DT_EXCLUSAO                          1699 non-null   datetime64[ns]
9   TIPO_PAGTO                           5342 non-null   object
10  NR_CONTRATO                          10272 non-null  int64
11  BASE                                 10272 non-null  object
12  NR_DIAS_ATRASO                       10272 non-null  int64
dtypes: datetime64[ns](2), float64(3), int64(2), object(6)
memory usage: 1.0+ MB
```

Figura 3 – Informações de registros e colunas.

São extraídas do conjunto de dados somente as colunas pertinentes à análise a ser realizada, portanto, em um primeiro momento, o conjunto de dados passa a ser composto apenas pelas colunas abaixo:

```
#-----#
# REMOVENDO COLUNAS QUE NÃO SERÃO UTILIZADAS: #
# CANAL, GRUPO, FLAG_MAIORES_IMPACTOS, NM_CLIENTE #
# DT_PAGTO, DT_EXCLUSAO, TIPO_PAGTO #
#-----#

df_sem_colunas_selecionadas = df.drop(["CANAL", "GRUPO", "FLAG_MAIORES_IMPACTOS",
                                       "NM_CLIENTE", "DT_PAGTO", "DT_EXCLUSAO", "TIPO_PAGTO"], axis = 1)
df_sem_colunas_selecionadas
```

	SALDO_PDD_POTENCIAL	SALDO_ESTOQUE_PDD	SALDO	NR_CONTRATO	BASE	NR_DIAS_ATRASO
0	1905.3300	211.7000	21170.29	7853468	SISABASE	56
1	178.5400	76.5100	2550.47	9721655	SISABASE	46
2	1489.7500	638.4700	21282.18	10834000	SISABASE	51
3	5043.7500	560.4200	18680.55	15361348	SISABASE	68
4	2451.9700	1225.9800	12259.83	20061412	SISABASE	71
...
10267	1882.7137	806.8773	26895.91	17021957	SISAENTR	47
10268	2142.1036	918.0444	30601.48	21314612	SISAENTR	47
10269	786.7538	393.3769	39337.69	23494354	SISAENTR	16

Figura 4 – Remoção de colunas que não serão utilizadas.

As colunas são posicionadas de forma que se entenda a estrutura da tabela.

```
#-----#
# ORDENAÇÃO DA POSIÇÃO DAS COLUNAS: #
#-----#

df_colunas_posicionadas = df_sem_colunas_selecionadas[["NR_CONTRATO", "BASE", "SALDO_PDD_POTENCIAL",
                                                         "SALDO_ESTOQUE_PDD", "SALDO", "NR_DIAS_ATRASO"]]
df_colunas_posicionadas
```

	NR_CONTRATO	BASE	SALDO_PDD_POTENCIAL	SALDO_ESTOQUE_PDD	SALDO	NR_DIAS_ATRASO
0	7853468	SISABASE	1905.3300	211.7000	21170.29	56
1	9721655	SISABASE	178.5400	76.5100	2550.47	46
2	10834000	SISABASE	1489.7500	638.4700	21282.18	51
3	15361348	SISABASE	5043.7500	560.4200	18680.55	68
4	20061412	SISABASE	2451.9700	1225.9800	12259.83	71
...
10267	17021957	SISAENTR	1882.7137	806.8773	26895.91	47
10268	21314612	SISAENTR	2142.1036	918.0444	30601.48	47
10269	23494354	SISAENTR	786.7538	393.3769	39337.69	16

Figura 5 – Ordenação da posição das colunas.

Ocorre a remoção de valores iguais ou menores do que zero, pois interferem na análise da cobrança. A coluna SALDO deve possuir valor positivo e maior do que zero, pois ela possui o total a ser efetuado pelo cliente. A coluba SALDO_PDD_POTENCIAL deve possuir um valor de parcela positivo e maior do que zero. E a coluna NR_DIAS_ATRASO deve possuir valores positivos maiores do que zero, pois representa os dias de atraso dos pagamentos.

```
#-----#
# REMOÇÃO DE VALORES NEGATIVOS OU ZERADOS: #
#-----#

# SÃO REMOVIDOS OS SALDOS COM VALORES MENORES DO QUE 1, POIS O BANCO NÃO FICA DEVENDO VALORES PARA OS CONTRATOS
df_saldo_positivo = df_colunas_posicionadas.loc[df_colunas_posicionadas['SALDO'] > 0]

# SÃO REMOVIDOS OS SALDO_PDD_POTENCIAL MENORES DO QUE 1, QUE SÃO AS PARCELAS, POIS SOMENTE PARCELAS EXISTENTES SÃO COBRADAS
df_saldo_pdd_potencial_posit = df_saldo_positivo.loc[df_saldo_positivo['SALDO_PDD_POTENCIAL'] > 0]

# SÃO REMOVIDOS OS DIAS EM ATRASO MENORES DO QUE 1, POIS SÃO COBRADOS SOMENTE CONTRATOS ATRASADOS
df_colunas_numeros_dias_atraso = df_saldo_pdd_potencial_posit.loc[df_saldo_pdd_potencial_posit['NR_DIAS_ATRASO'] > 0]
df_colunas_numeros_dias_atraso.count()
```

NR_CONTRATO	8925
BASE	8925
SALDO_PDD_POTENCIAL	8925
SALDO_ESTOQUE_PDD	8925
SALDO	8925
NR_DIAS_ATRASO	8925
dtype: int64	

Figura 6 – Remoção de valor negativos.

Foram identificados 157 contratos duplicados no dataframe.

```
# Contando registros duplicados na 'NR_CONTRATO'
registros_duplicados_contratos = df_colunas_numeros_dias_atraso['NR_CONTRATO'].duplicated().sum()
registros_duplicados_contratos
```

157

Figura 7 – Identificação de contratos duplicados.

Ordenação do dataframe pela coluna base. Os dados dessa coluna são SISABASE e SISAENTR.

SISABASE – dados que já existiam na base do banco;

SISAENTR – dados novos que entraram na base.

Até esta execução o total de registros é de 8925.

```
# Ordenação pela coluna BASE de forma ascendente
```

```
df_ordenado = df_colunas_numeros_dias_atraso.sort_values(by='BASE', ascending=False)
df_ordenado
```

	NR_CONTRATO	BASE	SALDO_PDD_POTENCIAL	SALDO_ESTOQUE_PDD	SALDO	NR_DIAS_ATRASO
10271	34645945	SISAENTR	23.76400	23.76400	4752.80	14
2182	40220899	SISAENTR	156.82720	78.41360	7841.36	29
4547	37395639	SISAENTR	11.38560	11.38560	2277.12	12
4546	605587302	SISAENTR	11.53065	11.53065	2306.13	14
4545	598536274	SISAENTR	50.46480	50.46480	10092.96	14
...
3111	94145836	SISABASE	4930.82000	2465.41000	24654.10	75
3110	89610943	SISABASE	1770.15000	758.64000	25287.94	46
3109	84003185	SISABASE	417.61000	208.81000	20880.63	43
3108	48772875	SISABASE	388.83000	166.64000	5554.66	33
4604	167592609	SISABASE	2900.19000	1450.09000	14500.93	76

8925 rows × 6 columns

Figura 8 – Ordenação pela coluna Base.

Na imagem abaixo está um exemplo de registros duplicados conforme a coluna contrato. Deverá permanecer o SALDO do registro que tem a coluna BASE igual a SISAENTR para esses duplicados.

```
# Exemplo de registros duplicados
```

```
contrato_selecionado = df_ordenado.query('NR_CONTRATO == 5901376')
contrato_selecionado
```

	NR_CONTRATO	BASE	SALDO_PDD_POTENCIAL	SALDO_ESTOQUE_PDD	SALDO	NR_DIAS_ATRASO
9881	5901376	SISAENTR	48.1998	48.1998	9639.96	7
1710	5901376	SISABASE	704.4700	301.9200	10063.90	37

Figura 9 – Exemplo de registros duplicados pela coluna NR_CONTRATO.

Para a remoção de contratos duplicados, tem de ser feita a unificação dos valores de alguns campos. O dataframe foi ordenado, previamente, pela coluna BASE onde foi feita por ordem alfabética e o SISAENTR vem em primeiro. Então são identificados os contratos duplicados pelo seu número identificador e são somados os seguintes campos: SALDO_PDD_POTENCIAL (valor das parcelas), SALDO_ESTOQUE_PDD (valor que já foi recuperado) e NR_DIAS_ATRASO (dias de atraso da parcela). Permanece o valor da coluna SALDO que está no registro do SISAENTR, pois pode ter ocorrido alguma alteração nesse campo (como desconto no saldo restante ou adição de valor).

Total de registros ficou em 8768.

```
# Identificação dos itens duplicados, soma das colunas SALDO_PDD_POTENCIAL, SALDO_ESTOQUE_PDD
# e NR_DIAS_ATRASO, mantendo o SALDO da base SISAENTR
df_somado = df_ordenado.groupby('NR_CONTRATO').agg({'BASE': 'first',
                                                    'SALDO_PDD_POTENCIAL': 'sum',
                                                    'SALDO_ESTOQUE_PDD': 'sum',
                                                    'SALDO': 'first',
                                                    'NR_DIAS_ATRASO': 'sum'}).reset_index()

df_somado
```

	NR_CONTRATO	BASE	SALDO_PDD_POTENCIAL	SALDO_ESTOQUE_PDD	SALDO	NR_DIAS_ATRASO
0	239202	SISAENTR	37.33680	37.33680	7467.36	11
1	1882463	SISABASE	222.35000	95.29000	3176.36	43
2	1948793	SISAENTR	8.69795	8.69795	1739.59	3
3	4678066	SISABASE	690.24000	295.82000	9860.57	33
4	5020136	SISABASE	1557.66000	667.57000	22252.28	48
...
8763	810898320	SISABASE	472.63000	236.32000	23631.61	22
8764	810909127	SISABASE	540.44000	231.61000	7720.47	49

Figura 10 – Remoção de duplicados pela coluna NR_CONTRATO.

Na imagem abaixo tem um exemplo de um contrato que estava duplicado e que foi removido de acordo com a coluna BASE, permanecendo o registro que possui o SISAENTR nessa coluna e somados os valores de alguns campos, permanecendo inalterado o campo SALDO para o registro SISAENTR.

```
# Exemplo do funcionamento de remoção de contrato duplicado
# e soma dos valores, com exceção do SALDO que permaneceu
# o valor de acordo com a coluna BASE SISAENTR
```

```
filtro = df_somado.query('NR_CONTRATO == 5901376')
filtro
```

	NR_CONTRATO	BASE	SALDO_PDD_POTENCIAL	SALDO_ESTOQUE_PDD	SALDO	NR_DIAS_ATRASO
13	5901376	SISAENTR	752.6698	350.1198	9639.96	44

Figura 10 – Exemplo de registro que tinha seu contrato duplicado e removido.

São criadas as colunas GRUPO e PARCELAS, que receberão valores de acordo com os outros campos da linha de registro. O preenchimento da coluna GRUPO será de acordo com o valor que estiver na coluna NR_DIAS_ATRASO e seguirá a seguinte regra:

- valor 30: para atrasos até 30 dias;
- valor 60: para atrasos de 31 dias até 60 dias;
- valor 90: para atrasos de 61 dias até 90 dias;
- valor 180: para atrasos de 91 dias até 180 dias;
- valor 365: para atrasos maiores de 180 dias.

A coluna PARCELAS será preenchida conforme a divisão da coluna SALDO com a coluna SALDO_PDD_POTENCIAL e o arredondamento desse total.

```
# Criação das colunas GRUPO e PARCELAS
# GRUPO agrupa os contratos pelos dias de atraso (até 30, 60, 90, 180 e 365)
# PARCELAS faz a divisão entre o SALDO devedor e parcela SALDO_PDD_POTENCIAL
# para estimar o número de parcelas restantes

df = df_somado
atraso = df['NR_DIAS_ATRASO']
df['GRUPO'] = np.where(atraso <= 30, 30, np.where((atraso > 30)
& (atraso <= 60), 60, np.where((atraso > 60)
& (atraso <= 90), 90, np.where((atraso > 90)
& (atraso <= 180), 180, 365))))

df['PARCELAS'] = (df['SALDO'] / df['SALDO_PDD_POTENCIAL']).round().astype(int)
df
```

	NR_CONTRATO	BASE	SALDO_PDD_POTENCIAL	SALDO_ESTOQUE_PDD	SALDO	NR_DIAS_ATRASO	GRUPO	PARCELAS
0	239202	SISAENTR	37.33680	37.33680	7467.36	11	30	200
1	1882463	SISABASE	222.35000	95.29000	3176.36	43	60	14
2	1948793	SISAENTR	8.69795	8.69795	1739.59	3	30	200
3	4678066	SISABASE	690.24000	295.82000	9860.57	33	60	14
4	5020136	SISABASE	1557.66000	667.57000	22252.28	48	60	14

Figura 11 – Criação das colunas GRUPO e PARCELAS.

São renomeadas todas as colunas para facilitar a visualização e entendimento. As principais mudanças dos nomes foram em:

Saldo_Potencial – valor da parcela pendente;

Saldo_Recuperado – valor recuperado (parcelas que foram pagas).

```
# Renomeação de colunas

df = df.rename(columns={'NR_CONTRATO': 'Contrato', 'NR_DIAS_ATRASO': 'Atraso', 'SALDO_ESTOQUE_PDD': 'Saldo_Recuperado'})
df = df.rename(columns={'SALDO_PDD_POTENCIAL': 'Saldo_Potencial', 'SALDO': 'Saldo', 'BASE': 'Base'})
df = df.rename(columns={'GRUPO': 'Grupo', 'PARCELAS': 'Parcelas'})
df
```

	Contrato	Base	Saldo_Potencial	Saldo_Recuperado	Saldo	Atraso	Grupo	Parcelas
0	239202	SISAENTR	37.33680	37.33680	7467.36	11	30	200
1	1882463	SISABASE	222.35000	95.29000	3176.36	43	60	14
2	1948793	SISAENTR	8.69795	8.69795	1739.59	3	30	200
3	4678066	SISABASE	690.24000	295.82000	9860.57	33	60	14
4	5020136	SISABASE	1557.66000	667.57000	22252.28	48	60	14
...
8763	810898320	SISABASE	472.63000	236.32000	23631.61	22	30	50
8764	810909127	SISABASE	540.44000	231.61000	7720.47	49	60	14

Figura 12 – Renomeação de todas as colunas.

4. Análise e Exploração dos Dados

Após tratado o conjunto de dados final, deve ser feita uma análise de frequência das coluna GRUPO para obter os totais dos valores distintos.

```
# Análise de frequência da coluna GRUPO

analise_frequencia = df['Grupo'].value_counts()
analise_frequencia
```

```
Grupo
60      3134
30      2674
90      1472
180     1307
365      181
Name: count, dtype: int64
```

Figura 13 – Análise de frequência da coluna GRUPO.

Com esse resultado sabemos que os grupo 60 e 30 dias tem a maior quantidade de contratos e que o grupo 365 tem a menor quantidade. Abaixo tem um gráfico de frequência da coluna Atraso para melhor visualização.

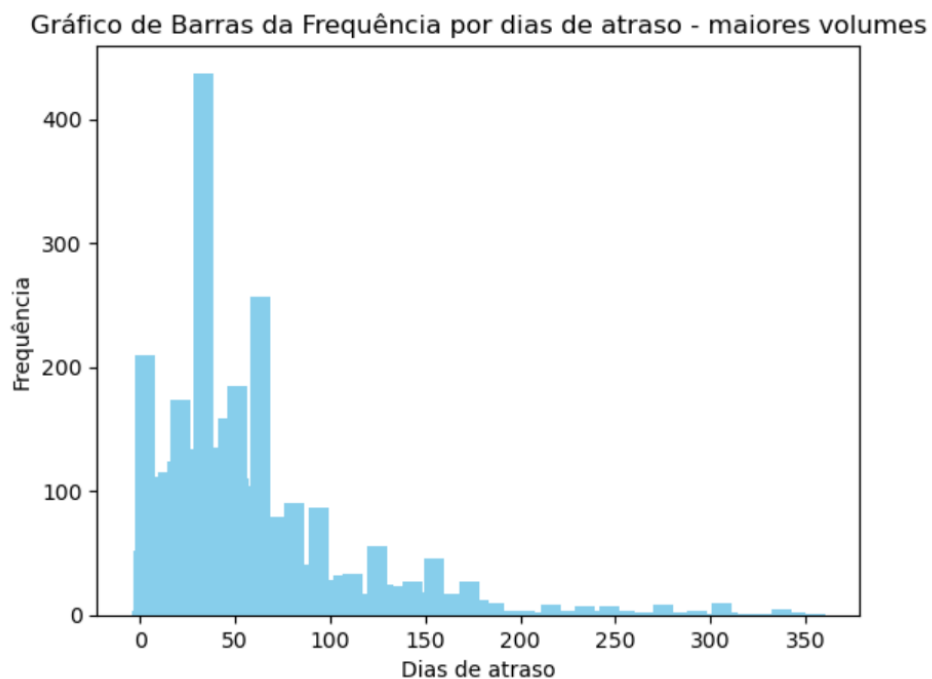


Figura 14 – Gráfico de barras da frequência da coluna Atraso.

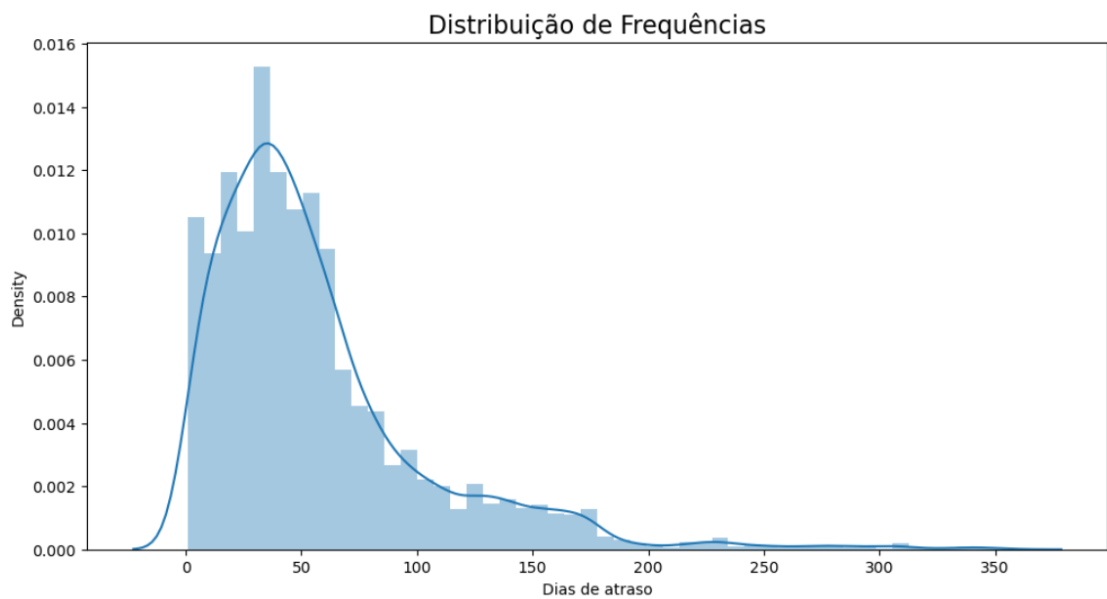


Figura 15 – Gráfico de barras da frequência com curva.

Continuando a análise dos dados no conjunto de dados tratados, ao plotar um boxplot, considerando todos os grupo e os valores de suas parcelas pendentes, temos o gráfico que segue.

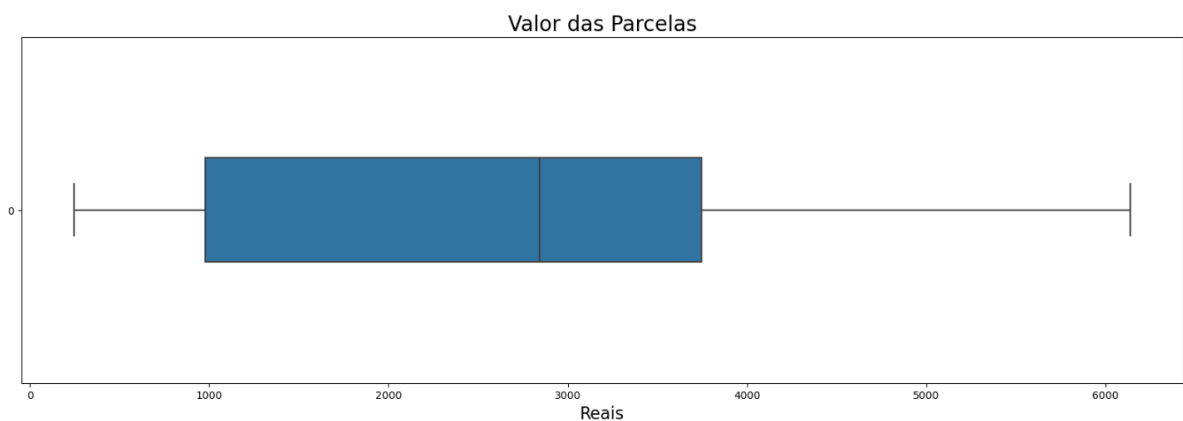


Figura 16 – Gráfico boxplot dos valores das parcelas.

Deste gráfico podemos interpretar que o valor da mediana entre os valores finais de parcelas em atraso está entre R\$ 2000,00 e R\$ 3000,00 e sem valores outliers multivariados naturais, ou seja, valores discrepantes dentro de um espaço multidimensional que ocorrem como exceções atípicas em relação ao conjunto de dados observado.

```
df['Saldo_Potencial'].describe().round(2)
```

count	8768.00
mean	1587.48
std	2419.96
min	2.15
25%	278.57
50%	769.32
75%	1962.32
max	46121.07

Name: Saldo_Potencial, dtype: float64

Figura 17 – Descrição da coluna Saldo_Potencial.

Na Figura 17 acima, temos os valores máximos, mínimos e os quartis da parcelas em atraso e que deverão ser cobradas.

Na figura seguinte está o saldo total das parcelas em atraso onde deverão ser feitas as análises para definir as formas de cobrança.

```
# Saldo total das parcelas em atraso
print('Total das parcelas em atraso: R$ ', df['Saldo_Potencial'].sum().round(2))
```

Total das parcelas em atraso: R\$ 13918991.61

Figura 18 – Total das parcelas em atraso.

Feita a análise por cada grupo, temos como analisar os dados individuais. O grupo 30 tem um total de 2674 contratos e valor de cobrança deste mês de R\$ 659045,22.

```
# Grupo 30

grupo_30 = df[df['Grupo'].eq(30)]
grupo_30['Saldo_Potencial'].describe().round(2)
```

count	2674.00
mean	246.46
std	218.42
min	2.15
25%	85.88
50%	184.35
75%	349.64
max	1796.22

Name: Saldo_Potencial, dtype: float64

```
# Saldo total das parcelas em atraso do grupo 30
saldo_potencial_grupo30 = grupo_30['Saldo_Potencial'].sum().round(2)
print('Total das parcelas em atraso: R$ ', saldo_potencial_grupo30)
```

Total das parcelas em atraso: R\$ 659045.22

Figura 19 – Descrição da coluna Saldo_Potencial do grupo 30.

O grupo 60 tem um total de 3134 contratos e valor de cobrança deste mês de R\$ 3066314,23.

```
# Grupo 60

grupo_60 = df[df['Grupo'].eq(60)]
grupo_60['Saldo_Potencial'].describe().round(2)

count    3134.00
mean      978.40
std       713.24
min       16.97
25%      496.42
50%      840.94
75%     1276.11
max      8427.12
Name: Saldo_Potencial, dtype: float64

# Saldo total das parcelas em atraso do grupo 60
saldo_potencial_grupo60 = grupo_60['Saldo_Potencial'].sum().round(2)
print ('Total das parcelas em atraso: R$ ', saldo_potencial_grupo60)

Total das parcelas em atraso: R$  3066314.23
```

Figura 20 – Descrição da coluna Saldo_Potencial do grupo 60.

O grupo 90 tem um total de 1472 contratos e valor de cobrança deste mês de R\$ 4184697,95.

```
# Grupo 90

grupo_90 = df[df['Grupo'].eq(90)]
grupo_90['Saldo_Potencial'].describe().round(2)

count    1472.00
mean     2842.87
std     2102.13
min       10.76
25%     1343.25
50%     2484.03
75%     3769.64
max     17965.13
Name: Saldo_Potencial, dtype: float64

# Saldo total das parcelas em atraso do grupo 90
saldo_potencial_grupo90 = grupo_90['Saldo_Potencial'].sum().round(2)
print ('Total das parcelas em atraso: R$ ', saldo_potencial_grupo90)

Total das parcelas em atraso: R$  4184697.95
```

Figura 21 – Descrição da coluna Saldo_Potencial do grupo 90.

O grupo 180 tem um total de 1307 contratos e valor de cobrança deste mês de R\$ 4898002,10.

```
# Grupo 180

grupo_180 = df[df['Grupo'].eq(180)]
grupo_180['Saldo_Potencial'].describe().round(2)

count      1307.00
mean       3747.51
std        3438.52
min         9.15
25%       1821.44
50%       3127.39
75%       4712.84
max       46093.73
Name: Saldo_Potencial, dtype: float64
```

```
# Saldo total das parcelas em atraso do grupo 180
saldo_potencial_grupo180 = grupo_180['Saldo_Potencial'].sum().round(2)
print('Total das parcelas em atraso: R$ ', saldo_potencial_grupo180)

Total das parcelas em atraso: R$  4898002.1
```

Figura 22 – Descrição da coluna Saldo_Potencial do grupo 180.

O grupo 365 tem um total de 181 contratos e valor de cobrança deste mês de R\$ 1110932,12.

```
# Grupo 365

grupo_365 = df[df['Grupo'].eq(365)]
grupo_365['Saldo_Potencial'].describe().round(2)

count      181.00
mean       6137.75
std       7280.64
min       186.10
25%       794.48
50%      4915.88
75%      8095.38
max     46121.07
Name: Saldo_Potencial, dtype: float64
```

```
# Saldo total das parcelas em atraso do grupo 365
saldo_potencial_grupo365 = grupo_365['Saldo_Potencial'].sum().round(2)
print('Total das parcelas em atraso: R$ ', saldo_potencial_grupo365)

Total das parcelas em atraso: R$  1110932.12
```

Figura 22 – Descrição da coluna Saldo_Potencial do grupo 365.

5. Criação de Modelos de Machine Learning

5.1 Estimando o Modelo

Para o problema apresentado, com o objetivo de calcular previsões quanto a recuperação de crédito veicular, foram utilizados alguns modelos e, para a aplicação dos modelos os dados foram divididos em conjuntos de treino e teste.

Os modelos foram divididos por grupos: 30, 60, 90, 180 e 365 dias, referentes aos atrasos. Para cada grupo foi feita uma média aritmética dos dias de atraso para identificar potenciais pagamentos, sendo que quanto menor o dia de atraso maior é o potencial da recuperação do crédito. Portanto, foi criada uma nova coluna chamada “Recuperar” que receberá o valor dessa estimativa. Ou seja, se os dias de atraso forem menores do que a média o valor do campo Recuperar será igual a 1 e os dias de atraso forem maiores ou iguais a média o valor do campo será igual a 0.

```
#-----#
# Média Aritmética para identificar potenciais pagamentos #
# Quanto menor o tempo no grupo, maior a chance de recuperação de crédito #
# Se os dias de atraso forem menores do que a média o valor do campo Recuperar será igual a 1 #
# Se os dias de atraso forem maiores ou iguais a média o valor do campo será igual a 0 #
#-----#

coluna_unica = grupo_30['Atraso'].drop_duplicates()
media_aritmetica = coluna_unica.mean()
media_aritmetica

grupo_30['Recuperar'] = grupo_30['Atraso'].apply(lambda x: 0 if x > media_aritmetica else (1 if x < media_aritmetica else 0))
```

Figura 23 – Descrição da coluna Recuperar e seus novos valores.

Para estimar o modelo, foram utilizadas duas variáveis, sendo uma a variável dependente (coluna “valor_Recuperar” do conjunto de dados) e a outra a contendo as variáveis independentes (“Saldo_Recuperado”, “Saldo” e “Atraso”).

```
#-----#
# VARIÁVEIS INDEPENDENTES #
#-----#
X = grupo_30[['Saldo_Recuperado', 'Saldo', 'Atraso']]
|
#-----#
# VARIÁVEL DEPENDENTE #
#-----#
y = grupo_30['Recuperar']
```

Figura 24 – Dividindo conjunto de dados em variáveis dependente e independentes.

A recuperação de crédito é frequentemente modelada usando técnicas de análise estatística, como regressão linear, regressão logística ou outros métodos de aprendizado de máquina, dependendo da complexidade do problema e da disponibilidade dos dados.

A regressão logística é um modelo estatístico utilizado para modelar a probabilidade de um evento ocorrer como função de um conjunto de variáveis independentes. Diferentemente da regressão linear, que prevê valores contínuos, a regressão logística é utilizada para problemas de classificação, onde a variável dependente é categórica e binária.

```
# Inicializar o modelo de regressão Logística
modelo = LogisticRegression()

# Treinar o modelo
modelo.fit(X_train_30, y_train_30)

# Fazer previsões no conjunto de teste
previsoes = modelo.predict(X_test_30)

# Avaliar a precisão do modelo
precisao = accuracy_score(y_test, previsoes)
print(f'Precisão do Modelo: {precisao}')
```

Precisão do Modelo: 0.5454545454545454

Figura 24 – Modelo de Regressão Logística

5.2 Linear Regression (Regressão Linear)

A regressão linear estuda a relação entre duas ou mais variáveis utilizando pontos de dados para encontrar a melhor linha de ajuste para modelagem dos dados. A equação $y = m * x + c$ representa a linha de melhor ajuste, onde y é a variável dependente e x é(são) a(s) variável(eis) independente(s), permitindo prever valores para determinada variável dependente com base em uma ou mais variáveis independentes.

Após a divisão das variáveis dependente e independentes, os dados foram separados em conjuntos de treino e teste através da função “train_test_split” da biblioteca “scikitlearn.model_selection”.

```
# 70% PARA TREINO
# 30% PARA TESTE
X_train_30, X_test_30, y_train_30, y_test_30 = train_test_split(X, y, test_size=0.3, random_state=42)
```

Figura 24 – Dividindo os dados em conjuntos de treino e teste com train_test_split.

O algoritmo é trabalhado e melhorado conforme lhe são apresentados dados, portanto, a divisão dos dados em conjuntos de treino e teste serve a este fim. A figura abaixo ilustra o caminho dos dados no processo de Machine Learning para treinar e testar um algoritmo.

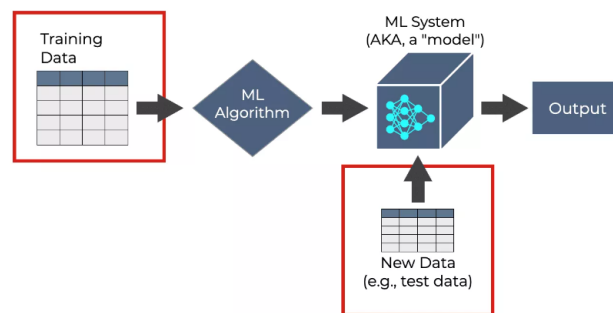


Figura 25 – Fases percorridas pelos dados em modelos de Machine Learning.

Desta forma, os conjuntos de dados foram estabelecidos com 70% dos dados para treino e 30% dos dados para teste. Nas etapas que seguem, o modelo de regressão linear é instanciado, ajustado e avaliado através de métricas e medidas.

5.2.1 Métrica MAE (Mean Absolute Error)

Calcula o erro absoluto médio entre os valores observados. Para esta métrica, quanto menor o valor obtido, melhor. A seguir, a fórmula utilizada para o referido cálculo.

The diagram shows the formula for Mean Absolute Error (MAE) with several annotations:

- A blue box around $\frac{1}{n}$ is labeled "Divide by the total number of data points".
- A green box around y is labeled "Actual output value".
- An orange box around \hat{y} is labeled "Predicted output value".
- A bracket under the absolute value term $|y - \hat{y}|$ is labeled "The absolute value of the residual".
- The summation symbol \sum is labeled "Sum of".

$$MAE = \frac{1}{n} \sum |y - \hat{y}|$$

Figura 26 – Fórmula para cálculo da métrica MAE.

5.2.2 Métrica RMSE (Root Mean Squared Error)

Calcula a raiz quadrática média de erros entre os valores observados (é a raiz quadrada do erro médio quadrado – MSE). Para esta métrica, quanto menor o valor obtido, melhor. A seguir, a fórmula utilizada para o referido cálculo.

The diagram shows the formula for Mean Squared Error (MSE) with several annotations:

- A bracket under n is labeled "test set".
- A bracket under y_i is labeled "predicted vaue" (note the typo).
- A bracket under \hat{y}_i is labeled "actual value".

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

Figura 27 – Fórmula para cálculo da métrica MSE.

5.2.3 R^2 ou Coeficiente de Determinação

Diz quanto o modelo está prevendo corretamente. O cálculo do R^2 envolve três medidas:

Medida	Descrição	Fórmula	Onde
Soma total dos quadrados (STQ)	Somatório das diferenças entre o valor alvo real e sua média elevada ao quadrado.	$\sum (y - \bar{y})^2$	y = valor real y traço = média
Soma dos quadrados dos resíduos (SQU)	Somatório das diferenças entre o valor predito e o valor real elevados ao quadrado.	$\sum (y - \hat{y})^2$	y = valor real y chapéu = valor predito
Soma dos quadrados de regressão (SQR)	Diferença entre o valor de SQT e SQU.	$SQR = SQT - SQU$	
R^2	Divisão da variação explicada pela variação total dos dados.	$R^2 = \frac{\sum (y - \bar{y})^2}{\sum (y - \hat{y})^2} = \frac{SQU}{SQT}$	

Figura 28 – Fórmulas de cálculo de R^2 ou Coeficiente de Determinação.

5.3 Regularização L1 (Lasso)

A regressão linear possui formas de regularizar sua função resultante com o objetivo de melhorar a relação de erros **bias** e **variance**. Um valor alto de **bias** indica que o modelo se ajusta pouco aos dados de treino, causando **underfitting** (neste caso, MSE (Mean Squared Error) é mais alto para o conjunto de dados de teste). Um valor alto de **variance** indica que o modelo se ajusta demais aos dados, causando **overfitting** (neste caso, MSE (Mean Squared Error) é zero para o conjunto de dados de teste).

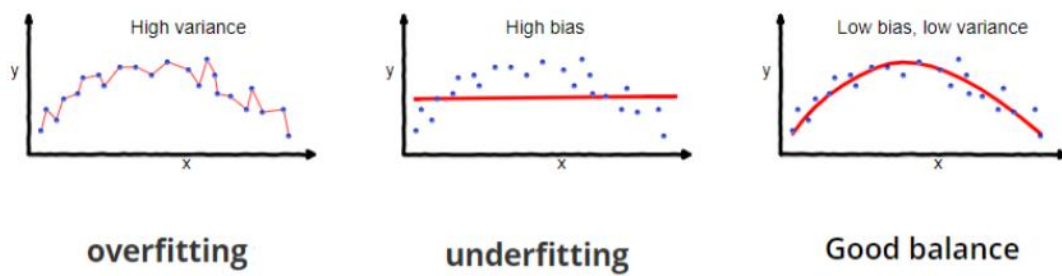


Figura 29 – Alternativas de ajuste dos modelos aos dados.

Com base nas informações acima podemos considerar que, quando temos:

Resultado	Indica
Baixo erro em dados de treino e alto erro em dados de teste	Valor alto de variância
Alto erro em dados de treino e erro parecido em dados de teste	Valor alto de bias
Alto erro em dados de treino e erro maior em dados de teste	Valor alto de bias Valor alto de variância
Baixo erro em dados de treino e baixo erro em dados de teste	Valor baixo de bias Valor baixo de variância

Figura 30 – Indicações de resultados para valores de bias e variância.

Lasso é uma das formas de como podemos regularizar a função através de penalidades. O procedimento de ajuste envolve a função de custo RSS (Residual Sum of Squares – Soma Residual dos Quadrados), que mede o nível de variância no termo de erro (ou resíduo) do modelo de regressão linear através da fórmula abaixo.

$$RSS = \sum_{i=1}^n [y_i - (\mathbf{w} \cdot \mathbf{x}_i + b)]^2$$

Figura 31 – Fórmula para cálculo da métrica RSS.

Quando ocorre *overfitting* em um modelo, ou seja, quando o modelo memoriza ruídos nos dados de treino, este fato está associado à variância do modelo, e uma forma de diminuir este erro é aumentando o *bias*. Para isso, são regularizados os coeficientes “w” que passam a ter tamanhos restritos. O processo

para este ajuste é adicionar um termo de regularização na função de custo para que os coeficientes sejam automaticamente minimizados.

$$\text{RSS}_{\text{lasso}} = \sum_{i=1}^n [y_i - (\mathbf{w} \cdot \mathbf{x}_i + b)]^2, \quad \boxed{+ \alpha \sum_{j=1}^p |w_j|}$$

regularização ℓ_1

Figura 32 – Fórmula para cálculo da métrica RSS com regularização L1 (Lasso).

Aplicando a regularização Lasso no modelo de regressão linear é realizada, automaticamente, a **feature selection** (seleção de recursos), que analisa as variáveis independentes verificando suas correlações que, caso sejam altas, implicam na seleção de apenas uma destas variáveis, gerando vários coeficientes com peso zero que são ignorados pelo modelo e facilitam sua interpretação, o que pode ser considerado vantajoso.

5.4 Regularização L2 (Ridge)

Outro ajuste que pode ser aplicado à regressão linear é o Ridge, ou L2, cuja regularização consiste na penalização da soma dos quadrados dos erros analisando dados que sofrem multicolinearidade (mínimos quadrados imparciais e altas variâncias, resultando em valores previsto bem distantes dos valores reais).

A regularização L2 é maior para coeficientes maiores, fazendo com que features correlacionadas tenham coeficientes parecidos. O termo de regularização abaixo é adicionado ao cálculo da função de custo.

Porém, isso não diminui a possibilidade da ocorrência de outliers no modelo, sendo recomendável limpar o conjunto de dados e remover as features desnecessárias antes da aplicação da regressão linear e dos ajustes (esta operação foi realizada em etapa anterior).

5.5 Regularização ElasticNet (L1 + L2)

A regularização ElasticNet consiste em encontrar e minimizar a soma dos quadrados dos erros, aplicando uma penalidade a esses coeficientes.

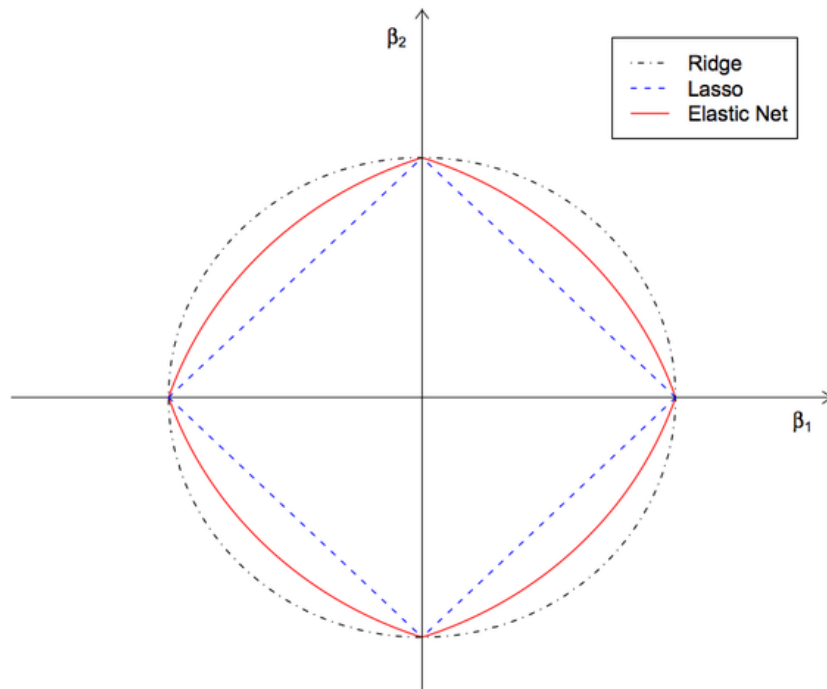


Figura 33 – Comportamento das regularizações Lasso, Ridge e ElasticNet.

Elasticnet combina as regularizações L1 (Lasso) e L2 (Ridge) para que seja obtido o melhor dos dois ajustes, realizando uma regularização mais eficiente através do cálculo representado pela fórmula abaixo.

$$RSS_{\text{elasticnet}} = \sum_{i=1}^n [y_i - (\mathbf{w} \cdot \mathbf{x}_i + b)]^2 + \alpha_1 \sum_{j=1}^p |w_j| + \alpha_2 \sum_{j=1}^p w_j^2$$

Figura 34 – Fórmula para cálculo da métrica RSS com regularizações L1 e L2 (ElasticNet).

Em comparação com a regularização Lasso (L1), ElasticNet amplia os limites de coleta de amostras para dados de alta dimensão, incluindo n variáveis até a saturação (em casos onde as variáveis são grupos altamente correlacionados, a regularização Lasso (L1) tende a uma variável deste grupo e ignorar completamente o restante).

5.6 Decision Tree (Árvore de Decisão)

As Árvores de Decisão são divididas em Árvores de Classificação e Árvores de Regressão. Como estamos lidando com uma variável de resposta numérica, serão utilizadas Árvores de Regressão.

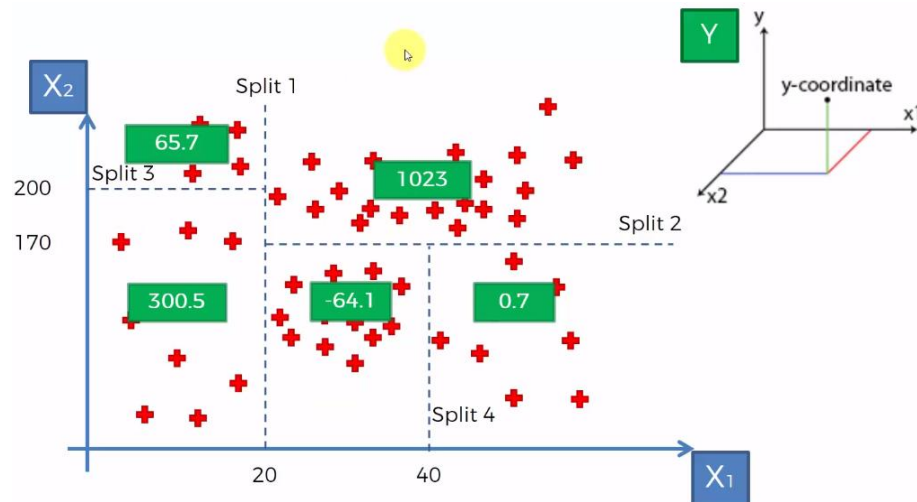


Figura 35 – Separação dos dados em conjuntos de dados menores.

O conjunto de dados é dividido em conjuntos cada vez menores (particionamento) e com instâncias de valores semelhantes, onde são observadas suas características. O modelo de Árvore de Regressão é construído de forma incremental, com o objetivo de produzir saídas contínuas significativas. O resultado é uma “árvore” com nós de decisão e nós folha. Cada nó de decisão possui duas ou mais ramificações representando valores para o atributo avaliado. O nó folha representa a decisão sobre o alvo numérico avaliado. O nó de decisão mais alto é chamado de nó raiz.

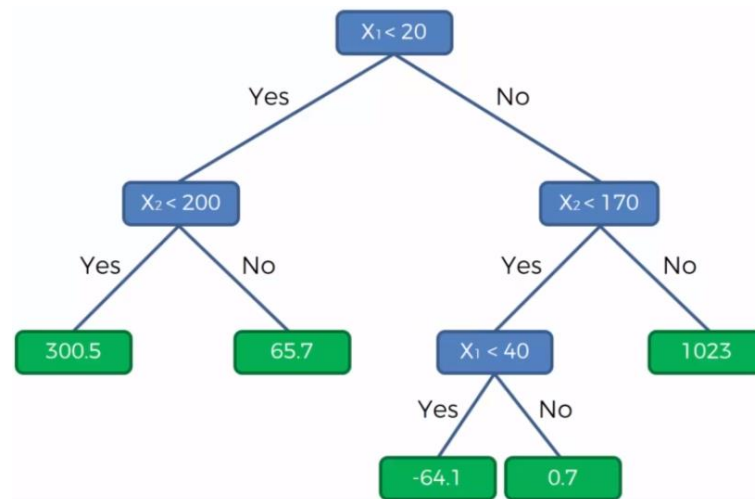


Figura 36 – Fluxograma gerado em uma Árvore de Decisão.

O algoritmo central utilizado pela Árvore de Decisão é chamado de ID3, que consiste na busca, de cima para baixo, de ramificações sem retrocesso. Em uma Árvore de Regressão o algoritmo ID3 pode substituir “Ganho de Informação” por “Redução de Desvio Padrão”.

O Desvio Padrão é utilizado para calcular a homogeneidade de uma amostra numérica. Se a amostra é completamente homogênea, o desvio padrão é zero.

$$\text{Standard Deviation} = S = \sqrt{\frac{\sum (x - \bar{x})^2}{n}}$$

Figura 37 – Fórmula para cálculo do Desvio Padrão.

A Redução do Desvio Padrão consiste em diminuir o valor do Desvio Padrão após a divisão do conjunto de dados em atributos. Construir uma Árvore de Decisão é como encontrar os ramos mais homogêneos para cada atributo.

5.7 Random Forest (Florestas Aleatórias)

O algoritmo de Florestas Aleatórias utiliza o método **bagging**, que consiste na combinação de diferentes modelos para se obter um único resultado, o tornando um algoritmo **ensambled** (que combina previsões de múltiplos algoritmos de Machine Learning), de forma a obter previsões mais acuradas do que modelos individuais.

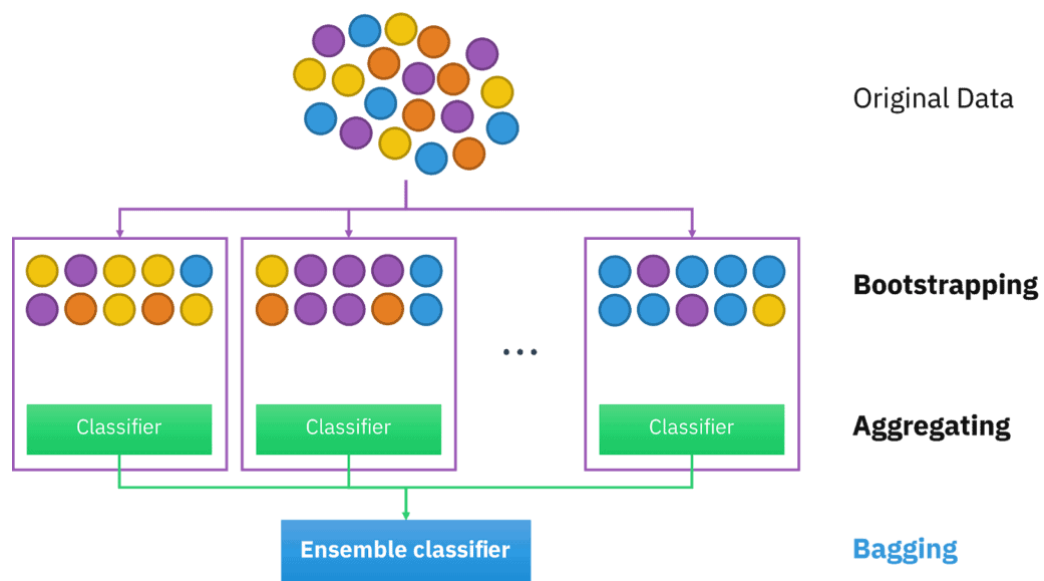


Figura 38 – Método Bagging.

É gerada uma estrutura de Árvores de Decisão, onde ocorre um treino para cada árvore em paralelo, ou seja, não ocorre interação entre elas, dando estabilidade ao algoritmo e reduzindo a variância. É um modelo que não se super-ajusta, devido à randomização de subconjuntos e recursos.

O algoritmo de Florestas Aleatórias possui, basicamente, quatro passos:

1. Seleção aleatória de algumas features
2. Seleção da feature mais adequada para a posição dono raiz
3. Geração dos nós filhos
4. Repetir os passos anteriores até atingir a quantidade de árvores desejada

6. Interpretação dos Resultados

6.1 Resultados das Métricas

Após a análise do conjunto de dados, foram construídos e aplicados os modelos de Machine Learning mencionados no tópico anterior e, para cada um deles foram avaliadas as métricas apresentadas no mesmo tópico.

Considerando o conjunto de dados por grupos, foram obtidos os resultados que seguem:

Modelo	Métricas		
	MAE	RMSE	R ²
Linear Regression	0.18	0.23	0.79
Lasso (L1)	0.22	0.27	0.70
Ridge (L2)	0.18	0.23	0.79
ElasticNet (L1 + L2)	0.19	0.24	0.76
Decision Tree	0.00	0.00	1.00
Random Forest	0.00	0.00	1.00

Figura 30 – Grupo 30

Com os valores obtidos para as métricas MAE, RMSE e R², pode-se concluir que os modelos de Machine Learning que tiveram melhor desempenho no grupo 30 foram Linear Regression (Regressão Linear) e Ridge (L2).

Modelo	Métricas		
	MAE	RMSE	R ²
Linear Regression	0.21	0.25	0.74
Lasso (L1)	0.23	0.28	0.68
Ridge (L2)	0.21	0.25	0.74
ElasticNet (L1 + L2)	0.21	0.26	0.72
Decision Tree	0.00	0.00	1.00
Random Forest	0.00	0.00	1.00

Figura 39 – Grupo 60

Com os valores obtidos para as métricas MAE, RMSE e R^2 , pode-se concluir que os modelos de Machine Learning que tiveram melhor desempenho no grupo 60 foram Linear Regression (Regressão Linear) e Ridge (L2).

Modelo	Métricas		
	MAE	RMSE	R^2
Linear Regression	0.19	0.24	0.75
Lasso (L1)	0.21	0.26	0.69
Ridge (L2)	0.19	0.24	0.75
ElasticNet (L1 + L2)	0.19	0.24	0.74
Decision Tree	0.00	0.00	1.00
Random Forest	0.00	0.00	1.00

Figura 40 – Grupo 90

Com os valores obtidos para as métricas MAE, RMSE e R^2 , pode-se concluir que os modelos de Machine Learning que tiveram melhor desempenho no grupo 90 foram Linear Regression (Regressão Linear) e Ridge (L2).

Modelo	Métricas		
	MAE	RMSE	R^2
Linear Regression	0.21	0.25	0.73
Lasso (L1)	0.21	0.26	0.72
Ridge (L2)	0.21	0.25	0.73
ElasticNet (L1 + L2)	0.21	0.25	0.73
Decision Tree	0.00	0.00	1.00
Random Forest	0.00	0.00	1.00

Figura 41 – Grupo 180

Com os valores obtidos para as métricas MAE, RMSE e R^2 , pode-se concluir que os modelos de Machine Learning que tiveram melhor desempenho no grupo 30 foram Linear Regression (Regressão Linear), Ridge (L2) e ElasticNet (L1 + L2).

Modelo	Métricas		
	MAE	RMSE	R ²
Linear Regression	0.20	0.24	-19.55
Lasso (L1)	0.19	0.25	-13.80
Ridge (L2)	0.20	0.24	0.74
ElasticNet (L1 + L2)	0.20	0.24	0.74
Decision Tree	0.00	0.00	1.00
Random Forest	0.00	0.02	1.00

Figura 42 – Grupo 365











Com os valores obtidos para as métricas MAE, RMSE e R², pode-se concluir que os modelos de Machine Learning que tiveram melhor desempenho no grupo 30 foram Ridge (L2) e ElasticNet (L1 + L2).

Assim como na avaliação tendo como alvo o conjunto de dados por inteiro, os resultados para os subconjuntos de dados separados por tipo de combustível mostram que os mesmos modelos, Linear Regression (Regressão Linear) e Ridge (L2) tiveram melhor desempenho.

7. Apresentação dos Resultados

A seguir, modelo Canvas proposto por Vesandani, para auxiliar e facilitar o planejamento estratégico do projeto de forma organizada e objetiva.

Nessa seção você deve apresentar os resultados obtidos. Apresente gráficos, *dashboards*, conte a sua história de forma bastante criativa. Aqui você pode utilizar os modelos de Canvas propostos por Dourard (clique [aqui](#)) ou por Vasandani (clique [aqui](#)).

The Machine Learning Canvas (v0.4) Designed for: Yuri Ramos Designed by: Yuri Ramos Date: Iteration:				
Decisions  How are predictions used to make decisions that provide the proposed value to the end-user? As previsões podem ser utilizadas para avaliar os contratos com maiores potenciais de recuperação de crédito após o atraso dos pagamentos.	ML task  Input, output to predict, type of problem. Entrada - Série mensal de contratos da base ou novos contratos. Saída - Previsão de potenciais recuperações de crédito	Value Propositions  What are we trying to do for the end-user(s) of the predictive system? What objectives are we serving? Estamos informando aos diretores, gerentes e mesas de cobrança os potenciais de recuperação de todos os contratos.	Data Sources  Which raw data sources can we use (internal and external)? O arquivo de dados privados utilizados é um xlsx disponibilizado pelo cliente bancário, ou seja, são dados sensíveis	Collecting Data  How do we get new data to learn from (inputs and outputs)? Para os novos treinos e testes, os dados continuam sendo disponibilizados pelo cliente diariamente e mensalmente.
Making Predictions  When do we make predictions on new inputs? How long do we have to featurize a new input and make a prediction? Previsões para novas entradas podem ser feitas através dos modelos de machine learning.	Offline Evaluation  Methods and metrics to evaluate the system before deployment. MAE (Mean Absolute Error) - Erro Absoluto Médio RMSE (Root Mean Squared Error) - Raiz Quadrática Média dos Erros R ² ou Coeficiente de Determinação	Features  Input representations extracted from raw data sources. [0,1] Onde: - 0 representa o menor potencial de recuperação - 1 representa o maior potencial de recuperação	Building Models  When do we create/update models with new training data? How long do we have to featurize training inputs and create a model? A partir do momento que os dados forem disponibilizados pelo cliente, coletados e tratados, eles podem ser processados como novos dados de treinamento, criando novos resultados para os modelos deste projeto.	
Live Evaluation and Monitoring  Methods and metrics to evaluate the system after deployment, and to quantify value creation. Podem ser utilizados os mesmos métodos e métricas utilizados na produção dos modelos de Machine Learning deste projeto.				

machinelearningcanvas.com by Louis Dorard, Ph.D. Licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

Figura 43 – Modelo Canvas do projeto.

8. Links

O material utilizado para este projeto está disponível no repositório do Git Hub através do link descrito abaixo. Através do link é possível ter acesso ao script (notebook) em linguagem Python que foi utilizado.

Para este projeto também foi preparada uma breve apresentação, trazendo as principais informações sobre ele. A apresentação se encontra disponível no link descrito abaixo.

Link para o repositório dos notebooks no GitHub:

<https://github.com/yurisavage/TCC>

Link para o vídeo da apresentação no YouTube:

<https://youtu.be/DfhatVLv-rc>

REFERÊNCIAS

GÉRON, Aurélien. **Mãos à Obra: Aprendizado de Máquina com Scikit-Learn & TensorFlow**. Rio de Janeiro: Alta Books, 2019.

GRUS, Joel. **Data Science do Zero**. Rio de Janeiro: Alta Books, 2016.

BHARGAVA, Aditya Y. **Entendendo Algoritmos**. São Paulo: Novatec, 2017.

<https://vemproitau.gupy.io/>

<https://veiculos.itaubr.com.br/atendimento/?acn>

APÊNDICE – Código do notebook utilizado no projeto

Notebook tcc_01_algoritmos_ml

```
#-----#
```

```
# CRIAR DIRETÓRIOS PARA DADOS E RESULTADOS #
```

```
#-----#
```

```
from pathlib import Path
```

```
from os import listdir
```

```
#-----#
```

```
# LISTAR E MANIPULAR ARQUIVOS DE DADOS #
```

```
#-----#
```

```
import pandas as pd
```

```
import os.path
```

```
from os.path import isfile, join
```

```
import random
```

```
import numpy as np
```

```
#-----#
```

```
# ANÁLISE DE DADOS - ALGORITMOS DE MACHINE LEARNING #
```

```
#-----#
```

```
# Machine Learning
```

```
from sklearn.model_selection import train_test_split
```

```
from sklearn import metrics, preprocessing
```

```
from sklearn.metrics import confusion_matrix, mean_absolute_error, mean_squared_error,  
accuracy_score
```

```
import statsmodels.api as sm
```

```
import scipy.stats as stats
```

```
import math
```

```

# Regressão Linear
from sklearn.linear_model import LinearRegression, Lasso, Ridge, ElasticNet,
LogisticRegression
from sklearn.datasets import make_regression

# Árvore de Decisão
from sklearn.tree import DecisionTreeRegressor

# Floresta Aleatória
from sklearn.ensemble import RandomForestRegressor

#-----#
# VISUALIZAÇÃO DE DADOS #
#-----#
import seaborn as sns
import matplotlib.pyplot as plt

#-----#
# INTERFACE GRÁFICA #
#-----#
#from PySimpleGUI import PySimpleGUI as sg

#-----#
# IGNORANDO WARNINGS #
#-----#
# Durante o desenvolvimento do notebooks foram exibidos warnings a respeito
# de bibliotecas que serão alteradas futuramente (pandas)
import warnings
warnings.filterwarnings('ignore')

```

```

#-----#
# SELECIONANDO ARQUIVO DE DADOS #
#-----#

#-----#
# Varrendo diretório onde está localizado o arquivo de dados #
#-----#

# Definindo nome do diretório
path = r"C:\TCC\00_tcc_dadoscoletados.xlsx"
path

df = pd.read_excel(path, engine='openpyxl')
df

#-----#
# IMPORTANDO REGISTROS DO ARQUIVO DE DADOS LISTADOS #
#-----#

# Verificar informações de total de registros e colunas
df.info()

#-----#
# REMOVENDO COLUNAS QUE NÃO SERÃO UTILIZADAS: #
# CANAL, GRUPO, FLAG_MAIORES_IMPACTOS, NM_CLIENTE #
# DT_PAGTO, DT_EXCLUSAO, TIPO_PAGTO #
#-----#

df_sem_colunas_selecionadas = df.drop(["CANAL", "GRUPO", "FLAG_MAIORES_IMPACTOS",
                                       "NM_CLIENTE", "DT_PAGTO", "DT_EXCLUSAO", "TIPO_PAGTO"], axis = 1)
df_sem_colunas_selecionadas

```

```
#-----#
```

```
# ORDENAÇÃO DA POSIÇÃO DAS COLUNAS: #
```

```
#-----#
```

```
df_colunas_posicionadas = df_sem_colunas_selecionadas[["NR_CONTRATO", "BASE",
"SALDO_PDD_POTENCIAL",
"SALDO_ESTOQUE_PDD", "SALDO", "NR_DIAS_ATRASO"]]
```

```
df_colunas_posicionadas
```

```
#-----#
```

```
# REMOÇÃO DE VALORES NEGATIVOS OU ZERADOS: #
```

```
#-----#
```

```
# SÃO REMOVIDOS OS SALDOS COM VALORES MENORES DO QUE 1, POIS O BANCO NÃO
FICA DEVENDO VALORES PARA OS CONTRATOS
```

```
df_saldo_positivo = df_colunas_posicionadas.loc[df_colunas_posicionadas['SALDO'] > 0]
```

```
# SÃO REMOVIDOS OS SALDO_PDD_POTENCIAL MENORES DO QUE 1, QUE SÃO AS
PARCELAS, POIS SOMENTE PARCELAS EXISTENTES SÃO COBRADAS
```

```
df_saldo_pdd_potencial_posit =
```

```
df_saldo_positivo.loc[df_saldo_positivo['SALDO_PDD_POTENCIAL'] > 0]
```

```
# SÃO REMOVIDOS OS DIAS EM ATRASO MENORES DO QUE 1, POIS SÃO COBRADOS
SOMENTE CONTRATOS ATRASADOS
```

```
df_colunas_numeros_dias_atraso =
```

```
df_saldo_pdd_potencial_posit.loc[df_saldo_pdd_potencial_posit['NR_DIAS_ATRASO'] > 0]
```

```
df_colunas_numeros_dias_atraso.count()
```

```
# Contando registros duplicados na 'NR_CONTRATO'
```

```
registros_duplicados_contratos
```

```
=
```

```
df_colunas_numeros_dias_atraso['NR_CONTRATO'].duplicated().sum()
```

```
registros_duplicados_contratos
```

```
# Ordenação pela coluna BASE de forma ascendente
```

```
df_ordenado = df_colunas_numeros_dias_atraso.sort_values(by='BASE', ascending=False)
```

```
df_ordenado
```

```
# Exemplo de registros duplicados
```

```
contrato_selecionado = df_ordenado.query('NR_CONTRATO == 5901376')
```

```
contrato_selecionado
```

```
# Identificação dos itens duplicados, soma das colunas SALDO_PDD_POTENCIAL,  
SALDO_ESTOQUE_PDD
```

```
# e NR_DIAS_ATRASO, mantendo o SALDO da base SISAENTR
```

```
df_somado = df_ordenado.groupby('NR_CONTRATO').agg({'BASE': 'first',  
                                                    'SALDO_PDD_POTENCIAL': 'sum',  
                                                    'SALDO_ESTOQUE_PDD': 'sum',  
                                                    'SALDO': 'first',  
                                                    'NR_DIAS_ATRASO': 'sum'}).reset_index()
```

```
df_somado
```

```
# Exemplo do funcionamento de remoção de contrato duplicado
```

```
# e soma dos valores, com exceção do SALDO que permaneceu
```

```
# o valor de acordo com a coluna BASE SISAENTR
```

```
filtro = df_somado.query('NR_CONTRATO == 5901376')
```

```
filtro
```

```

# Criação das colunas GRUPO e PARCELAS
# GRUPO agrupa os contratos pelos dias de atraso (até 30, 60, 90, 180 e 365)
# PARCELAS faz a divisão entre o SALDO devedor e parcela SALDO_PDD_POTENCIAL
# para estimar o número de parcelas restantes

df = df_somado
atraso = df['NR_DIAS_ATRASO']
df['GRUPO'] = np.where(atraso <= 30, 30, np.where((atraso > 30)
        & (atraso <= 60), 60, np.where((atraso > 60)
        & (atraso <= 90), 90, np.where((atraso > 90)
        & (atraso <= 180), 180, 365))))

df['PARCELAS'] = (df['SALDO'] / df['SALDO_PDD_POTENCIAL']).round().astype(int)

df

# Renomeação de colunas

df = df.rename(columns={'NR_CONTRATO': 'Contrato', 'NR_DIAS_ATRASO': 'Atraso',
'SALDO_ESTOQUE_PDD': 'Saldo_Recuperado'})
df = df.rename(columns={'SALDO_PDD_POTENCIAL': 'Saldo_Potencial', 'SALDO': 'Saldo',
'BASE': 'Base'})
df = df.rename(columns={'GRUPO': 'Grupo', 'PARCELAS': 'Parcelas'})
df

# Análise de frequência da coluna GRUPO

analise_frequencia = df['Grupo'].value_counts()
analise_frequencia

```

```
# Gráfico das quantidades de contratos por dias de atraso
```

```
frequencia = df['Atraso'].value_counts()
```

```
frequencia
```

```
tabela_frequencia = pd.DataFrame({'Atraso': frequencia.index, 'Frequencia':  
frequencia.values})
```

```
plt.bar(tabela_frequencia['Atraso'], tabela_frequencia['Frequencia'], color='skyblue',  
width=10.5)
```

```
# Adicionar rótulos e título
```

```
plt.xlabel('Dias de atraso')
```

```
plt.ylabel('Frequência')
```

```
plt.title('Gráfico de Barras da Frequência por dias de atraso - maiores volumes')
```

```
# Exibir o gráfico
```

```
plt.show()
```

```
# Agrupando dados por Grupo e Saldo_Potencial, obtendo a média do atraso
```

```
df_agg = df.groupby(['Grupo', 'Saldo_Potencial']).agg({'Atraso' : 'mean'}).reset_index()
```

```
df_agg
```

```
# Curva sobre a distribuição de Frequências
```

```
ax = sns.distplot(df_agg['Atraso'])
```

```
ax.figure.set_size_inches(12, 6)
```

```
ax.set_title('Distribuição de Frequências', fontsize = 16)
```

```
ax.set_xlabel('Dias de Atraso')
```

```
# Agrupando dados por Grupo, obtendo a média do valor das parcelas atrasadas
```

```
df_agg = df.groupby(['Grupo']).agg({'Saldo_Potencial' : 'mean'}).reset_index()
```

```
# Boxplot utilizando valores agrupados
```

```
ax = sns.boxplot(data = df_agg['Saldo_Potencial'], orient = 'h', width = 0.3)
```

```
ax.figure.set_size_inches(20, 6)
```

```
ax.set_title('Valor das Parcelas', fontsize=20)
```

```
ax.set_xlabel('Reais', fontsize=16)
```

```
ax
```

```
# Descrição da coluna Saldo_Potencial
```

```
df['Saldo_Potencial'].describe().round(2)
```

```
# Saldo total das parcelas em atraso
```

```
print ('Total das parcelas em atraso: R$ ', df['Saldo_Potencial'].sum().round(2))
```

```
# Grupo 30
```

```
grupo_30 = df[df['Grupo'].eq(30)]
```

```
grupo_30['Saldo_Potencial'].describe().round(2)
```

```
# Saldo total das parcelas em atraso do grupo 30
```

```
saldo_potencial_grupo30 = grupo_30['Saldo_Potencial'].sum().round(2)
```

```
print ('Total das parcelas em atraso: R$ ', saldo_potencial_grupo30)
```

```
# Grupo 60
```

```
grupo_60 = df[df['Grupo'].eq(60)]
```

```
grupo_60['Saldo_Potencial'].describe().round(2)
```

```
# Saldo total das parcelas em atraso do grupo 60
```

```
saldo_potencial_grupo60 = grupo_60['Saldo_Potencial'].sum().round(2)
```

```
print ('Total das parcelas em atraso: R$ ', saldo_potencial_grupo60)
```


Grupo 90

```
grupo_90 = df[df['Grupo'].eq(90)]
grupo_90['Saldo_Potencial'].describe().round(2)
```

Saldo total das parcelas em atraso do grupo 90

```
saldo_potencial_grupo90 = grupo_90['Saldo_Potencial'].sum().round(2)
print ('Total das parcelas em atraso: R$ ', saldo_potencial_grupo90)
```

Grupo 180

```
grupo_180 = df[df['Grupo'].eq(180)]
grupo_180['Saldo_Potencial'].describe().round(2)
```

Saldo total das parcelas em atraso do grupo 180

```
saldo_potencial_grupo180 = grupo_180['Saldo_Potencial'].sum().round(2)
print ('Total das parcelas em atraso: R$ ', saldo_potencial_grupo180)
```

Grupo 365

```
grupo_365 = df[df['Grupo'].eq(365)]
grupo_365['Saldo_Potencial'].describe().round(2)
```

Saldo total das parcelas em atraso do grupo 365

```
saldo_potencial_grupo365 = grupo_365['Saldo_Potencial'].sum().round(2)
print ('Total das parcelas em atraso: R$ ', saldo_potencial_grupo365)
```

#-----#

Média Aritmética para identificar potenciais pagamentos

Quanto menor o tempo no grupo, maior a chance de recuperação de crédito

Se os dias de atraso forem menores do que a média o valor do campo Recuperar será igual a 1

```
# Se os dias de atraso forem maiores ou iguais a média o valor do campo será igual a 0 #
```

```
#-----#
```

```
coluna_unica = grupo_30['Atraso'].drop_duplicates()
```

```
media_aritmetica = coluna_unica.mean()
```

```
media_aritmetica
```

```
grupo_30['Recuperar'] = grupo_30['Atraso'].apply(lambda x: 0 if x > media_aritmetica else (1  
if x < media_aritmetica else 0))
```

```
# Separar as variáveis independentes (X) e a variável dependente (y)
```

```
#-----#
```

```
# VARIÁVEIS INDEPENDENTES #
```

```
#-----#
```

```
X = grupo_30[['Saldo_Recuperado', 'Saldo', 'Atraso']]
```

```
#-----#
```

```
# VARIÁVEL DEPENDENTE #
```

```
#-----#
```

```
y = grupo_30['Recuperar']
```

```
#-----#
```

```
# DIVIDINDO DADOS EM CONJUNTOS DE TREINO E TESTE #
```

```
#-----#
```

```
# 70% PARA TREINO
```

```
# 30% PARA TESTE
```

```
X_train_30, X_test_30, y_train_30, y_test_30 = train_test_split(X, y, test_size=0.3,  
random_state=42)
```

```
# Inicializar o modelo de regressão logística
modelo = LogisticRegression()
```

```
# Treinar o modelo
modelo.fit(X_train_30, y_train_30)
```

```
# Fazer previsões no conjunto de teste
previsoes = modelo.predict(X_test_30)
```

```
# Avaliar a precisão do modelo
precisao = accuracy_score(y_test, previsoes)
print(f'Precisão do Modelo: {precisao}')
```

```
#-----#
```

```
# INSTANCIANDO MODELOS #
```

```
#-----#
```

```
# Linear Regression
modelo_lr_30 = LinearRegression()
```

```
# Lasso (L1)
modelo_lasso_30 = Lasso()
```

```
# Ridge (L2)
modelo_ridge_30 = Ridge()
```

```
# ElasticNet (L1 + L2)
modelo_elasticnet_30 = ElasticNet()
```

```
# Decision Tree Regressor
modelo_dt_30 = DecisionTreeRegressor()
```

```
# Random Forest Regressor
modelo_rf_30 = RandomForestRegressor()
```

```
#-----#
```

```
# AJUSTANDO MODELOS #
```

```
#-----#
```

```
# Treinar o modelo
modelo_lr_30.fit(X_train_30, y_train_30)
```

```
# Lasso (L1)
modelo_lasso_30 = Lasso()
```

```
# Ridge (L2)
modelo_ridge_30 = Ridge()
```

```
# ElasticNet (L1 + L2)
modelo_elasticnet_30 = ElasticNet()
```

```
# Decision Tree Regressor
modelo_dt_30 = DecisionTreeRegressor()
```

```
# Random Forest Regressor
modelo_rf_30 = RandomForestRegressor()
```

```
#-----#
```

```
# AJUSTANDO MODELOS #
```

```
#-----#
```

```
# Linear Regression
```

```
modelo_lr_30.fit(X_train_30, y_train_30)
```

```
# Lasso (L1)
```

```
modelo_lasso_30.fit(X_train_30, y_train_30)
```

```
# Ridge (L2)
```

```
modelo_ridge_30.fit(X_train_30, y_train_30)
```

```
# ElasticNet (L1 + L2)
```

```
modelo_elasticnet_30.fit(X_train_30, y_train_30)
```

```
# Decision Tree Regressor
```

```
modelo_dt_30.fit(X_train_30, y_train_30)
```

```
# Random Forest Regressor
```

```
modelo_rf_30.fit(X_train_30, y_train_30)
```

```
#-----#
```

```
# MÉTRICAS PARA DADOS DE TREINO #
```

```
#-----#
```

```
#-----#
```

```
# Definindo um valor previsto #
```

```
#-----#
```

```
# Linear Regression
```

```
y_previsto_train_lr_30 = modelo_lr_30.predict(X_train_30)
```

```
# Lasso (L1)
```

```
y_previsto_train_lasso_30 = modelo_lasso_30.predict(X_train_30)
```

```

# Ridge (L2)
y_previsto_train_ridge_30 = modelo_ridge_30.predict(X_train_30)

# ElasticNet (L1 + L2)
y_previsto_train_elasticnet_30 = modelo_elasticnet_30.predict(X_train_30)

# Decision Tree Regressor
y_previsto_train_dt_30 = modelo_dt_30.predict(X_train_30)

# Random Forest Regressor
y_previsto_train_rf_30 = modelo_rf_30.predict(X_train_30)

#-----#
# Calculando métricas #
#-----#

#-----#
# Linear Regression #
#-----#
print('-----')
print('--    LINEAR REGRESSION    --')
print('-----')

modelo_mae_train_lr_30 = mean_absolute_error(y_train_30, y_previsto_train_lr_30)
print('MAE Treino = {0:.2f}'.format(modelo_mae_train_lr_30))

modelo_mse_train_lr_30 = mean_squared_error(y_train_30, y_previsto_train_lr_30)
modelo_rmse_train_lr_30 = math.sqrt(modelo_mse_train_lr_30)
print('RMSE Treino = {0:.2f}'.format(modelo_rmse_train_lr_30))

modelo_r2_train_lr_30 = modelo_lr_30.score(X_train_30, y_train_30)
print('R² Treino = {0:.2f}'.format(modelo_r2_train_lr_30))

```

```

#-----#
# Ajuste Lasso (L1) #
#-----#
print('-----')
print('--  AJUSTE LASSO (L1)  --')
print('-----')

modelo_mae_train_lasso_30 = mean_absolute_error(y_train_30,
y_previsto_train_lasso_30)
print('MAE Treino = {0:.2f}'.format(modelo_mae_train_lasso_30))

modelo_mse_train_lasso_30 = mean_squared_error(y_train_30, y_previsto_train_lasso_30)
modelo_rmse_train_lasso_30 = math.sqrt(modelo_mse_train_lasso_30)
print('RMSE Treino = {0:.2f}'.format(modelo_rmse_train_lasso_30))

modelo_r2_train_lasso_30 = modelo_lasso_30.score(X_train_30, y_train_30)
print('R² Treino = {0:.2f}'.format(modelo_r2_train_lasso_30))

#-----#
# Ajuste Ridge (L2) #
#-----#
print('-----')
print('--  AJUSTE RIDGE (L2)  --')
print('-----')

modelo_mae_train_ridge_30 = mean_absolute_error(y_train_30,
y_previsto_train_ridge_30)
print('MAE Treino = {0:.2f}'.format(modelo_mae_train_ridge_30))

modelo_mse_train_ridge_30 = mean_squared_error(y_train_30, y_previsto_train_ridge_30)
modelo_rmse_train_ridge_30 = math.sqrt(modelo_mse_train_ridge_30)

```

```

print('RMSE Treino = {0:.2f}'.format(modelo_rmse_train_ridge_30))

modelo_r2_train_ridge_30 = modelo_ridge_30.score(X_train_30, y_train_30)
print('R² Treino = {0:.2f}'.format(modelo_r2_train_ridge_30))

#-----#
# Ajuste ElasticNet (L1 + L2) #
#-----#
print('-----')
print('-- AJUSTE ELASTICNET (L1 + L2) --')
print('-----')

modelo_mae_train_elasticnet_30 = mean_absolute_error(y_train_30,
y_previsto_train_elasticnet_30)
print('MAE Treino = {0:.2f}'.format(modelo_mae_train_elasticnet_30))

modelo_mse_train_elasticnet_30 = mean_squared_error(y_train_30,
y_previsto_train_elasticnet_30)
modelo_rmse_train_elasticnet_30 = math.sqrt(modelo_mse_train_elasticnet_30)
print('RMSE Treino = {0:.2f}'.format(modelo_rmse_train_elasticnet_30))

modelo_r2_train_elasticnet_30 = modelo_elasticnet_30.score(X_train_30, y_train_30)
print('R² Treino = {0:.2f}'.format(modelo_r2_train_elasticnet_30))

#-----#
# Decision Tree Regressor #
#-----#
print('-----')
print('-- DECISION TREE REGRESSOR --')
print('-----')

modelo_mae_train_dt_30 = mean_absolute_error(y_train_30, y_previsto_train_dt_30)

```



```

print('MAE Treino = {0:.2f}'.format(modelo_mae_train_dt_30))

modelo_mse_train_dt_30 = mean_squared_error(y_train_30, y_previsto_train_dt_30)
modelo_rmse_train_dt_30 = math.sqrt(modelo_mse_train_dt_30)
print('RMSE Treino = {0:.2f}'.format(modelo_rmse_train_dt_30))

modelo_r2_train_dt_30 = modelo_dt_30.score(X_train_30, y_train_30)
print('R² Treino = {0:.2f}'.format(modelo_r2_train_dt_30))

#-----#
# Random Forest Regressor #
#-----#
print('-----')
print('-- RANDOM FOREST REGRESSOR --')
print('-----')

modelo_mae_train_rf_30 = mean_absolute_error(y_train_30, y_previsto_train_rf_30)
print('MAE Treino = {0:.2f}'.format(modelo_mae_train_rf_30))

modelo_mse_train_rf_30 = mean_squared_error(y_train_30, y_previsto_train_rf_30)
modelo_rmse_train_rf_30 = math.sqrt(modelo_mse_train_rf_30)
print('RMSE Treino = {0:.2f}'.format(modelo_rmse_train_rf_30))

modelo_r2_train_rf_30 = modelo_rf_30.score(X_train_30, y_train_30)
print('R² Treino = {0:.2f}'.format(modelo_r2_train_rf_30))

#-----#
# MÉTRICAS PARA DADOS DE TESTE #
#-----#

#-----#

```

```

# Definindo um valor previsto #
#-----#

# Linear Regression
y_previsto_test_lr_30 = modelo_lr_30.predict(X_test_30)

# Lasso (L1)
y_previsto_test_lasso_30 = modelo_lasso_30.predict(X_test_30)

# Ridge (L2)
y_previsto_test_ridge_30 = modelo_ridge_30.predict(X_test_30)

# ElasticNet (L1 + L2)
y_previsto_test_elasticnet_30 = modelo_elasticnet_30.predict(X_test_30)

# Decision Tree Regressor
y_previsto_test_dt_30 = modelo_dt_30.predict(X_test_30)

# Random Forest Regressor
y_previsto_test_rf_30 = modelo_rf_30.predict(X_test_30)

#-----#
# Calculando métricas #
#-----#

#-----#
# Linear Regression #
#-----#
print('-----')
print('--    LINEAR REGRESSION    --')
print('-----')

```

```

modelo_mae_test_lr_30 = mean_absolute_error(y_test_30, y_previsto_test_lr_30)
print('MAE Teste = {0:.2f}'.format(modelo_mae_test_lr_30))

```

```

modelo_mse_test_lr_30 = mean_squared_error(y_test_30, y_previsto_test_lr_30)
modelo_rmse_test_lr_30 = math.sqrt(modelo_mse_test_lr_30)
print('RMSE Teste = {0:.2f}'.format(modelo_rmse_test_lr_30))

```

```

modelo_r2_test_lr_30 = modelo_lr_30.score(X_test_30, y_test_30)
print('R2 Teste = {0:.2f}'.format(modelo_r2_test_lr_30))

```

```

#-----#
# Ajuste Lasso (L1) #
#-----#
print('-----')
print('--    AJUSTE LASSO (L1)    --')
print('-----')

```

```

modelo_mae_test_lasso_30 = mean_absolute_error(y_test_30, y_previsto_test_lasso_30)
print('MAE Teste = {0:.2f}'.format(modelo_mae_test_lasso_30))

```

```

modelo_mse_test_lasso_30 = mean_squared_error(y_test_30, y_previsto_test_lasso_30)
modelo_rmse_test_lasso_30 = math.sqrt(modelo_mse_test_lasso_30)
print('RMSE Teste = {0:.2f}'.format(modelo_rmse_test_lasso_30))

```

```

modelo_r2_test_lasso_30 = modelo_lasso_30.score(X_test_30, y_test_30)
print('R2 Teste = {0:.2f}'.format(modelo_r2_test_lasso_30))

```

```

#-----#
# Ajuste Ridge (L2) #
#-----#
print('-----')
print('--    AJUSTE RIDGE (L2)    --')

```

```
print('-----')
```

```
modelo_mae_test_ridge_30 = mean_absolute_error(y_test_30, y_previsto_test_ridge_30)
```

```
print('MAE Teste = {0:.2f}'.format(modelo_mae_test_ridge_30))
```

```
modelo_mse_test_ridge_30 = mean_squared_error(y_test_30, y_previsto_test_ridge_30)
```

```
modelo_rmse_test_ridge_30 = math.sqrt(modelo_mse_test_ridge_30)
```

```
print('RMSE Teste = {0:.2f}'.format(modelo_rmse_test_ridge_30))
```

```
modelo_r2_test_ridge_30 = modelo_ridge_30.score(X_test_30, y_test_30)
```

```
print('R² Teste = {0:.2f}'.format(modelo_r2_test_ridge_30))
```

```
#-----#
```

```
# Ajuste ElasticNet (L1 + L2) #
```

```
#-----#
```

```
print('-----')
```

```
print('-- AJUSTE ELASTICNET (L1 + L2) --')
```

```
print('-----')
```

```
modelo_mae_test_elasticnet_30 = mean_absolute_error(y_test_30,
y_previsto_test_elasticnet_30)
```

```
print('MAE Teste = {0:.2f}'.format(modelo_mae_test_elasticnet_30))
```

```
modelo_mse_test_elasticnet_30 = mean_squared_error(y_test_30,
y_previsto_test_elasticnet_30)
```

```
modelo_rmse_test_elasticnet_30 = math.sqrt(modelo_mse_test_elasticnet_30)
```

```
print('RMSE Teste = {0:.2f}'.format(modelo_rmse_test_elasticnet_30))
```

```
modelo_r2_test_elasticnet_30 = modelo_elasticnet_30.score(X_test_30, y_test_30)
```

```
print('R² Teste = {0:.2f}'.format(modelo_r2_test_elasticnet_30))
```

```
#-----#
```

```

# Decision Tree Regressor #
#-----#
print('-----')
print('-- DECISION TREE REGRESSOR --')
print('-----')

modelo_mae_test_dt_30 = mean_absolute_error(y_test_30, y_previsto_test_dt_30)
print('MAE Teste = {0:.2f}'.format(modelo_mae_test_dt_30))

modelo_mse_test_dt_30 = mean_squared_error(y_test_30, y_previsto_test_dt_30)
modelo_rmse_test_dt_30 = math.sqrt(modelo_mse_test_dt_30)
print('RMSE Teste = {0:.2f}'.format(modelo_rmse_test_dt_30))

modelo_r2_test_dt_30 = modelo_dt_30.score(X_test_30, y_test_30)
print('R² Teste = {0:.2f}'.format(modelo_r2_test_dt_30))

#-----#
# Random Forest Regressor #
#-----#
print('-----')
print('-- RANDOM FOREST REGRESSOR --')
print('-----')

modelo_mae_test_rf_30 = mean_absolute_error(y_test_30, y_previsto_test_rf_30)
print('MAE Teste = {0:.2f}'.format(modelo_mae_test_rf_30))

modelo_mse_test_rf_30 = mean_squared_error(y_test_30, y_previsto_test_rf_30)
modelo_rmse_test_rf_30 = math.sqrt(modelo_mse_test_rf_30)
print('RMSE Teste = {0:.2f}'.format(modelo_rmse_test_rf_30))

modelo_r2_test_rf_30 = modelo_rf_30.score(X_test_30, y_test_30)
print('R² Teste = {0:.2f}'.format(modelo_r2_test_rf_30))

```

```

#-----#
# Média Aritmética para identificar potenciais pagamentos #
# Quanto menor o tempo no grupo, maior a chance de recuperação de crédito #
# Se os dias de atraso forem menores do que a média o valor do campo Recuperar será igual
a 1 #
# Se os dias de atraso forem maiores ou iguais a média o valor do campo será igual a 0 #
#-----#

coluna_unica = grupo_60['Atraso'].drop_duplicates()
media_aritmetica = coluna_unica.mean()
media_aritmetica

grupo_60['Recuperar'] = grupo_60['Atraso'].apply(lambda x: 0 if x > media_aritmetica else (1
if x < media_aritmetica else 0))

# Separar as variáveis independentes (X) e a variável dependente (y)
#-----#
# VARIÁVEIS INDEPENDENTES #
#-----#
X = grupo_60[['Saldo_Recuperado', 'Saldo', 'Atraso']]

#-----#
# VARIÁVEL DEPENDENTE #
#-----#
y = grupo_60['Recuperar']

#-----#
# DIVIDINDO DADOS EM CONJUNTOS DE TREINO E TESTE #
#-----#

```

```

# 70% PARA TREINO
# 30% PARA TESTE
X_train_60, X_test_60, y_train_60, y_test_60 = train_test_split(X, y, test_size=0.3,
random_state=42)

# Inicializar o modelo de regressão logística
modelo = LogisticRegression()

# Treinar o modelo
modelo.fit(X_train_60, y_train_60)

# Fazer previsões no conjunto de teste
previsoes = modelo.predict(X_test_60)

# Avaliar a precisão do modelo
precisao = accuracy_score(y_test_60, previsoes)
print(f'Precisão do Modelo: {precisao}')

#-----#
# INSTANCIANDO MODELOS #
#-----#

# Linear Regression
modelo_lr_60 = LinearRegression()

# Lasso (L1)
modelo_lasso_60 = Lasso()

# Ridge (L2)
modelo_ridge_60 = Ridge()

```

```
# ElasticNet (L1 + L2)
modelo_elasticnet_60 = ElasticNet()

# Decision Tree Regressor
modelo_dt_60 = DecisionTreeRegressor()

# Random Forest Regressor
modelo_rf_60 = RandomForestRegressor()

#-----#
# AJUSTANDO MODELOS #
#-----#

# Treinar o modelo
modelo_lr_60.fit(X_train_60, y_train_60)

# Lasso (L1)
modelo_lasso_60 = Lasso()

# Ridge (L2)
modelo_ridge_60 = Ridge()

# ElasticNet (L1 + L2)
modelo_elasticnet_60 = ElasticNet()

# Decision Tree Regressor
modelo_dt_60 = DecisionTreeRegressor()

# Random Forest Regressor
modelo_rf_60 = RandomForestRegressor()
```



```
#-----#
```

```
# AJUSTANDO MODELOS #
```

```
#-----#
```

```
# Linear Regression
```

```
modelo_lr_60.fit(X_train_60, y_train_60)
```

```
# Lasso (L1)
```

```
modelo_lasso_60.fit(X_train_60, y_train_60)
```

```
# Ridge (L2)
```

```
modelo_ridge_60.fit(X_train_60, y_train_60)
```

```
# ElasticNet (L1 + L2)
```

```
modelo_elasticnet_60.fit(X_train_60, y_train_60)
```

```
# Decision Tree Regressor
```

```
modelo_dt_60.fit(X_train_60, y_train_60)
```

```
# Random Forest Regressor
```

```
modelo_rf_60.fit(X_train_60, y_train_60)
```

```
#-----#
```

```
# MÉTRICAS PARA DADOS DE TREINO #
```

```
#-----#
```

```
#-----#
```

```
# Definindo um valor previsto #
```

```
#-----#
```

```
# Linear Regression
```

```

y_previsto_train_lr_60 = modelo_lr_60.predict(X_train_60)

# Lasso (L1)
y_previsto_train_lasso_60 = modelo_lasso_60.predict(X_train_60)

# Ridge (L2)
y_previsto_train_ridge_60 = modelo_ridge_60.predict(X_train_60)

# ElasticNet (L1 + L2)
y_previsto_train_elasticnet_60 = modelo_elasticnet_60.predict(X_train_60)

# Decision Tree Regressor
y_previsto_train_dt_60 = modelo_dt_60.predict(X_train_60)

# Random Forest Regressor
y_previsto_train_rf_60 = modelo_rf_60.predict(X_train_60)

#-----#
# Calculando métricas #
#-----#

#-----#
# Linear Regression #
#-----#
print('-----')
print('--    LINEAR REGRESSION    --')
print('-----')

modelo_mae_train_lr_60 = mean_absolute_error(y_train_60, y_previsto_train_lr_60)
print('MAE Treino = {0:.2f}'.format(modelo_mae_train_lr_60))

modelo_mse_train_lr_60 = mean_squared_error(y_train_60, y_previsto_train_lr_60)

```

```

modelo_rmse_train_lr_60 = math.sqrt(modelo_mse_train_lr_60)
print('RMSE Treino = {0:.2f}'.format(modelo_rmse_train_lr_60))

```

```

modelo_r2_train_lr_60 = modelo_lr_60.score(X_train_60, y_train_60)
print('R² Treino = {0:.2f}'.format(modelo_r2_train_lr_60))

```

```

#-----#
# Ajuste Lasso (L1) #
#-----#
print('-----')
print('-- AJUSTE LASSO (L1) --')
print('-----')

```

```

modelo_mae_train_lasso_60 = mean_absolute_error(y_train_60,
y_previsto_train_lasso_60)
print('MAE Treino = {0:.2f}'.format(modelo_mae_train_lasso_60))

```

```

modelo_mse_train_lasso_60 = mean_squared_error(y_train_60, y_previsto_train_lasso_60)
modelo_rmse_train_lasso_60 = math.sqrt(modelo_mse_train_lasso_60)
print('RMSE Treino = {0:.2f}'.format(modelo_rmse_train_lasso_60))

```

```

modelo_r2_train_lasso_60 = modelo_lasso_60.score(X_train_60, y_train_60)
print('R² Treino = {0:.2f}'.format(modelo_r2_train_lasso_60))

```

```

#-----#
# Ajuste Ridge (L2) #
#-----#
print('-----')
print('-- AJUSTE RIDGE (L2) --')
print('-----')

```

```

modelo_mae_train_ridge_60          =          mean_absolute_error(y_train_60,
y_previsto_train_ridge_60)
print('MAE Treino = {0:.2f}'.format(modelo_mae_train_ridge_60))

```

```

modelo_mse_train_ridge_60 = mean_squared_error(y_train_60, y_previsto_train_ridge_60)
modelo_rmse_train_ridge_60 = math.sqrt(modelo_mse_train_ridge_60)
print('RMSE Treino = {0:.2f}'.format(modelo_rmse_train_ridge_60))

```

```

modelo_r2_train_ridge_60 = modelo_ridge_60.score(X_train_60, y_train_60)
print('R² Treino = {0:.2f}'.format(modelo_r2_train_ridge_60))

```

```

#-----#
# Ajuste ElasticNet (L1 + L2) #
#-----#
print('-----')
print('-- AJUSTE ELASTICNET (L1 + L2) --')
print('-----')

```

```

modelo_mae_train_elasticnet_60      =          mean_absolute_error(y_train_60,
y_previsto_train_elasticnet_60)
print('MAE Treino = {0:.2f}'.format(modelo_mae_train_elasticnet_60))

```

```

modelo_mse_train_elasticnet_60      =          mean_squared_error(y_train_60,
y_previsto_train_elasticnet_60)
modelo_rmse_train_elasticnet_60 = math.sqrt(modelo_mse_train_elasticnet_60)
print('RMSE Treino = {0:.2f}'.format(modelo_rmse_train_elasticnet_60))

```

```

modelo_r2_train_elasticnet_60 = modelo_elasticnet_60.score(X_train_60, y_train_60)
print('R² Treino = {0:.2f}'.format(modelo_r2_train_elasticnet_60))

```

```

#-----#
# Decision Tree Regressor #

```

```

#-----#
print('-----')
print('-- DECISION TREE REGRESSOR --')
print('-----')

modelo_mae_train_dt_60 = mean_absolute_error(y_train_60, y_previsto_train_dt_60)
print('MAE Treino = {0:.2f}'.format(modelo_mae_train_dt_60))

modelo_mse_train_dt_60 = mean_squared_error(y_train_60, y_previsto_train_dt_60)
modelo_rmse_train_dt_60 = math.sqrt(modelo_mse_train_dt_60)
print('RMSE Treino = {0:.2f}'.format(modelo_rmse_train_dt_60))

modelo_r2_train_dt_60 = modelo_dt_60.score(X_train_60, y_train_60)
print('R² Treino = {0:.2f}'.format(modelo_r2_train_dt_60))

#-----#
# Random Forest Regressor #
#-----#
print('-----')
print('-- RANDOM FOREST REGRESSOR --')
print('-----')

modelo_mae_train_rf_60 = mean_absolute_error(y_train_60, y_previsto_train_rf_60)
print('MAE Treino = {0:.2f}'.format(modelo_mae_train_rf_60))

modelo_mse_train_rf_60 = mean_squared_error(y_train_60, y_previsto_train_rf_60)
modelo_rmse_train_rf_60 = math.sqrt(modelo_mse_train_rf_60)
print('RMSE Treino = {0:.2f}'.format(modelo_rmse_train_rf_60))

modelo_r2_train_rf_60 = modelo_rf_60.score(X_train_60, y_train_60)
print('R² Treino = {0:.2f}'.format(modelo_r2_train_rf_60))

```

```

#-----#
# MÉTRICAS PARA DADOS DE TESTE #
#-----#

#-----#
# Definindo um valor previsto #
#-----#

# Linear Regression
y_previsto_test_lr_60 = modelo_lr_60.predict(X_test_60)

# Lasso (L1)
y_previsto_test_lasso_60 = modelo_lasso_60.predict(X_test_60)

# Ridge (L2)
y_previsto_test_ridge_60 = modelo_ridge_60.predict(X_test_60)

# ElasticNet (L1 + L2)
y_previsto_test_elasticnet_60 = modelo_elasticnet_60.predict(X_test_60)

# Decision Tree Regressor
y_previsto_test_dt_60 = modelo_dt_60.predict(X_test_60)

# Random Forest Regressor
y_previsto_test_rf_60 = modelo_rf_60.predict(X_test_60)

#-----#
# Calculando métricas #
#-----#

#-----#
# Linear Regression #

```

```

#-----#
print('-----')
print('--    LINEAR REGRESSION    --')
print('-----')

modelo_mae_test_lr_60 = mean_absolute_error(y_test_60, y_previsto_test_lr_60)
print('MAE Teste = {0:.2f}'.format(modelo_mae_test_lr_60))

modelo_mse_test_lr_60 = mean_squared_error(y_test_60, y_previsto_test_lr_60)
modelo_rmse_test_lr_60 = math.sqrt(modelo_mse_test_lr_60)
print('RMSE Teste = {0:.2f}'.format(modelo_rmse_test_lr_60))

modelo_r2_test_lr_60 = modelo_lr_60.score(X_test_60, y_test_60)
print('R² Teste = {0:.2f}'.format(modelo_r2_test_lr_60))

#-----#
# Ajuste Lasso (L1) #
#-----#
print('-----')
print('--    AJUSTE LASSO (L1)    --')
print('-----')

modelo_mae_test_lasso_60 = mean_absolute_error(y_test_60, y_previsto_test_lasso_60)
print('MAE Teste = {0:.2f}'.format(modelo_mae_test_lasso_60))

modelo_mse_test_lasso_60 = mean_squared_error(y_test_60, y_previsto_test_lasso_60)
modelo_rmse_test_lasso_60 = math.sqrt(modelo_mse_test_lasso_60)
print('RMSE Teste = {0:.2f}'.format(modelo_rmse_test_lasso_60))

modelo_r2_test_lasso_60 = modelo_lasso_60.score(X_test_60, y_test_60)
print('R² Teste = {0:.2f}'.format(modelo_r2_test_lasso_60))

```

```

#-----#
# Ajuste Ridge (L2) #
#-----#

print('-----')
print('-- AJUSTE RIDGE (L2) --')
print('-----')

modelo_mae_test_ridge_60 = mean_absolute_error(y_test_60, y_previsto_test_ridge_60)
print('MAE Teste = {0:.2f}'.format(modelo_mae_test_ridge_60))

modelo_mse_test_ridge_60 = mean_squared_error(y_test_60, y_previsto_test_ridge_60)
modelo_rmse_test_ridge_60 = math.sqrt(modelo_mse_test_ridge_60)
print('RMSE Teste = {0:.2f}'.format(modelo_rmse_test_ridge_60))

modelo_r2_test_ridge_60 = modelo_ridge_60.score(X_test_30, y_test_30)
print('R² Teste = {0:.2f}'.format(modelo_r2_test_ridge_60))

#-----#
# Ajuste ElasticNet (L1 + L2) #
#-----#

print('-----')
print('-- AJUSTE ELASTICNET (L1 + L2) --')
print('-----')

modelo_mae_test_elasticnet_60 = mean_absolute_error(y_test_60,
y_previsto_test_elasticnet_60)
print('MAE Teste = {0:.2f}'.format(modelo_mae_test_elasticnet_60))

modelo_mse_test_elasticnet_60 = mean_squared_error(y_test_60,
y_previsto_test_elasticnet_60)
modelo_rmse_test_elasticnet_60 = math.sqrt(modelo_mse_test_elasticnet_60)
print('RMSE Teste = {0:.2f}'.format(modelo_rmse_test_elasticnet_60))

```



```

modelo_r2_test_elasticnet_60 = modelo_elasticnet_60.score(X_test_60, y_test_60)
print('R2 Teste = {0:.2f}'.format(modelo_r2_test_elasticnet_60))

```

```

#-----#

```

```

# Decision Tree Regressor #

```

```

#-----#

```

```

print('-----')

```

```

print('-- DECISION TREE REGRESSOR --')

```

```

print('-----')

```

```

modelo_mae_test_dt_60 = mean_absolute_error(y_test_60, y_previsto_test_dt_60)
print('MAE Teste = {0:.2f}'.format(modelo_mae_test_dt_60))

```

```

modelo_mse_test_dt_60 = mean_squared_error(y_test_60, y_previsto_test_dt_60)
modelo_rmse_test_dt_60 = math.sqrt(modelo_mse_test_dt_60)
print('RMSE Teste = {0:.2f}'.format(modelo_rmse_test_dt_60))

```

```

modelo_r2_test_dt_60 = modelo_dt_60.score(X_test_60, y_test_60)
print('R2 Teste = {0:.2f}'.format(modelo_r2_test_dt_60))

```

```

#-----#

```

```

# Random Forest Regressor #

```

```

#-----#

```

```

print('-----')

```

```

print('-- RANDOM FOREST REGRESSOR --')

```

```

print('-----')

```

```

modelo_mae_test_rf_60 = mean_absolute_error(y_test_60, y_previsto_test_rf_60)
print('MAE Teste = {0:.2f}'.format(modelo_mae_test_rf_60))

```

```

modelo_mse_test_rf_60 = mean_squared_error(y_test_60, y_previsto_test_rf_60)

```

```

modelo_rmse_test_rf_60 = math.sqrt(modelo_mse_test_rf_60)
print('RMSE Teste = {0:.2f}'.format(modelo_rmse_test_rf_60))

```

```

modelo_r2_test_rf_60 = modelo_rf_60.score(X_test_60, y_test_60)
print('R² Teste = {0:.2f}'.format(modelo_r2_test_rf_60))

```

```

#-----#

```

```

# Média Aritmética para identificar potenciais pagamentos #

```

```

# Quanto menor o tempo no grupo, maior a chance de recuperação de crédito #

```

```

# Se os dias de atraso forem menores do que a média o valor do campo Recuperar será igual
a 1 #

```

```

# Se os dias de atraso forem maiores ou iguais a média o valor do campo será igual a 0 #

```

```

#-----#

```

```

coluna_unica = grupo_90['Atraso'].drop_duplicates()

```

```

media_aritmetica = coluna_unica.mean()

```

```

media_aritmetica

```

```

grupo_90['Recuperar'] = grupo_90['Atraso'].apply(lambda x: 0 if x > media_aritmetica else (1
if x < media_aritmetica else 0))

```

```

# Separar as variáveis independentes (X) e a variável dependente (y)

```

```

#-----#

```

```

# VARIÁVEIS INDEPENDENTES #

```

```

#-----#

```

```

X = grupo_90[['Saldo_Recuperado', 'Saldo', 'Atraso']]

```

```

#-----#

```

```

# VARIÁVEL DEPENDENTE #

```

```

#-----#
y = grupo_90['Recuperar']

#-----#
# DIVIDINDO DADOS EM CONJUNTOS DE TREINO E TESTE #
#-----#

# 70% PARA TREINO
# 30% PARA TESTE
X_train_90, X_test_90, y_train_90, y_test_90 = train_test_split(X, y, test_size=0.3,
random_state=42)

# Inicializar o modelo de regressão logística
modelo = LogisticRegression()

# Treinar o modelo
modelo.fit(X_train_90, y_train_90)

# Fazer previsões no conjunto de teste
previsoes = modelo.predict(X_test_90)

# Avaliar a precisão do modelo
precisao = accuracy_score(y_test_90, previsoes)
print(f'Precisão do Modelo: {precisao}')

#-----#
# INSTANCIANDO MODELOS #
#-----#

# Linear Regression
modelo_lr_90 = LinearRegression()

```

```
# Lasso (L1)
```

```
modelo_lasso_90 = Lasso()
```

```
# Ridge (L2)
```

```
modelo_ridge_90 = Ridge()
```

```
# ElasticNet (L1 + L2)
```

```
modelo_elasticnet_90 = ElasticNet()
```

```
# Decision Tree Regressor
```

```
modelo_dt_90 = DecisionTreeRegressor()
```

```
# Random Forest Regressor
```

```
modelo_rf_90 = RandomForestRegressor()
```

```
#-----#
```

```
# AJUSTANDO MODELOS #
```

```
#-----#
```

```
# Treinar o modelo
```

```
modelo_lr_90.fit(X_train_90, y_train_90)
```

```
# Lasso (L1)
```

```
modelo_lasso_90 = Lasso()
```

```
# Ridge (L2)
```

```
modelo_ridge_90 = Ridge()
```

```
# ElasticNet (L1 + L2)
```

```
modelo_elasticnet_90 = ElasticNet()
```

```

# Decision Tree Regressor
modelo_dt_90 = DecisionTreeRegressor()

# Random Forest Regressor
modelo_rf_90 = RandomForestRegressor()

#-----#
# AJUSTANDO MODELOS #
#-----#

# Linear Regression
modelo_lr_90.fit(X_train_90, y_train_90)

# Lasso (L1)
modelo_lasso_90.fit(X_train_90, y_train_90)

# Ridge (L2)
modelo_ridge_90.fit(X_train_90, y_train_90)

# ElasticNet (L1 + L2)
modelo_elasticnet_90.fit(X_train_90, y_train_90)

# Decision Tree Regressor
modelo_dt_90.fit(X_train_90, y_train_90)

# Random Forest Regressor
modelo_rf_90.fit(X_train_90, y_train_90)

#-----#
# MÉTRICAS PARA DADOS DE TREINO #
#-----#

```

```

#-----#
# Definindo um valor previsto #
#-----#

# Linear Regression
y_previsto_train_lr_90 = modelo_lr_90.predict(X_train_90)

# Lasso (L1)
y_previsto_train_lasso_90 = modelo_lasso_90.predict(X_train_90)

# Ridge (L2)
y_previsto_train_ridge_90 = modelo_ridge_90.predict(X_train_90)

# ElasticNet (L1 + L2)
y_previsto_train_elasticnet_90 = modelo_elasticnet_90.predict(X_train_90)

# Decision Tree Regressor
y_previsto_train_dt_90 = modelo_dt_90.predict(X_train_90)

# Random Forest Regressor
y_previsto_train_rf_90 = modelo_rf_90.predict(X_train_90)

#-----#
# Calculando métricas #
#-----#

#-----#
# Linear Regression #
#-----#
print('-----')
print('--    LINEAR REGRESSION    --')
print('-----')

```

```

modelo_mae_train_lr_90 = mean_absolute_error(y_train_90, y_previsto_train_lr_90)
print('MAE Treino = {0:.2f}'.format(modelo_mae_train_lr_90))

```

```

modelo_mse_train_lr_90 = mean_squared_error(y_train_90, y_previsto_train_lr_90)
modelo_rmse_train_lr_90 = math.sqrt(modelo_mse_train_lr_90)
print('RMSE Treino = {0:.2f}'.format(modelo_rmse_train_lr_90))

```

```

modelo_r2_train_lr_90 = modelo_lr_90.score(X_train_90, y_train_90)
print('R² Treino = {0:.2f}'.format(modelo_r2_train_lr_90))

```

```

#-----#
# Ajuste Lasso (L1) #
#-----#
print('-----')
print('-- AJUSTE LASSO (L1) --')
print('-----')

```

```

modelo_mae_train_lasso_90 = mean_absolute_error(y_train_90,
y_previsto_train_lasso_90)
print('MAE Treino = {0:.2f}'.format(modelo_mae_train_lasso_90))

```

```

modelo_mse_train_lasso_90 = mean_squared_error(y_train_90, y_previsto_train_lasso_90)
modelo_rmse_train_lasso_90 = math.sqrt(modelo_mse_train_lasso_90)
print('RMSE Treino = {0:.2f}'.format(modelo_rmse_train_lasso_90))

```

```

modelo_r2_train_lasso_90 = modelo_lasso_90.score(X_train_90, y_train_90)
print('R² Treino = {0:.2f}'.format(modelo_r2_train_lasso_90))

```

```

#-----#
# Ajuste Ridge (L2) #
#-----#

```

```

print('-----')
print('-- AJUSTE RIDGE (L2) --')
print('-----')

modelo_mae_train_ridge_90 = mean_absolute_error(y_train_90,
y_previsto_train_ridge_90)
print('MAE Treino = {0:.2f}'.format(modelo_mae_train_ridge_90))

modelo_mse_train_ridge_90 = mean_squared_error(y_train_90, y_previsto_train_ridge_90)
modelo_rmse_train_ridge_90 = math.sqrt(modelo_mse_train_ridge_90)
print('RMSE Treino = {0:.2f}'.format(modelo_rmse_train_ridge_90))

modelo_r2_train_ridge_90 = modelo_ridge_90.score(X_train_90, y_train_90)
print('R2 Treino = {0:.2f}'.format(modelo_r2_train_ridge_90))

#-----#
# Ajuste ElasticNet (L1 + L2) #
#-----#
print('-----')
print('-- AJUSTE ELASTICNET (L1 + L2) --')
print('-----')

modelo_mae_train_elasticnet_90 = mean_absolute_error(y_train_90,
y_previsto_train_elasticnet_90)
print('MAE Treino = {0:.2f}'.format(modelo_mae_train_elasticnet_90))

modelo_mse_train_elasticnet_90 = mean_squared_error(y_train_90,
y_previsto_train_elasticnet_90)
modelo_rmse_train_elasticnet_90 = math.sqrt(modelo_mse_train_elasticnet_90)
print('RMSE Treino = {0:.2f}'.format(modelo_rmse_train_elasticnet_90))

modelo_r2_train_elasticnet_90 = modelo_elasticnet_90.score(X_train_90, y_train_90)

```



```

print('R2 Treino = {0:.2f}'.format(modelo_r2_train_elasticnet_90))

#-----#
# Decision Tree Regressor #
#-----#
print('-----')
print('-- DECISION TREE REGRESSOR --')
print('-----')

modelo_mae_train_dt_90 = mean_absolute_error(y_train_90, y_previsto_train_dt_90)
print('MAE Treino = {0:.2f}'.format(modelo_mae_train_dt_90))

modelo_mse_train_dt_90 = mean_squared_error(y_train_90, y_previsto_train_dt_90)
modelo_rmse_train_dt_90 = math.sqrt(modelo_mse_train_dt_90)
print('RMSE Treino = {0:.2f}'.format(modelo_rmse_train_dt_90))

modelo_r2_train_dt_90 = modelo_dt_90.score(X_train_90, y_train_90)
print('R2 Treino = {0:.2f}'.format(modelo_r2_train_dt_90))

#-----#
# Random Forest Regressor #
#-----#
print('-----')
print('-- RANDOM FOREST REGRESSOR --')
print('-----')

modelo_mae_train_rf_90 = mean_absolute_error(y_train_90, y_previsto_train_rf_90)
print('MAE Treino = {0:.2f}'.format(modelo_mae_train_rf_90))

modelo_mse_train_rf_90 = mean_squared_error(y_train_90, y_previsto_train_rf_90)
modelo_rmse_train_rf_90 = math.sqrt(modelo_mse_train_rf_90)
print('RMSE Treino = {0:.2f}'.format(modelo_rmse_train_rf_90))

```

```

modelo_r2_train_rf_90 = modelo_rf_90.score(X_train_90, y_train_90)
print('R² Treino = {0:.2f}'.format(modelo_r2_train_rf_90))

#-----#
# MÉTRICAS PARA DADOS DE TESTE #
#-----#

#-----#
# Definindo um valor previsto #
#-----#

# Linear Regression
y_previsto_test_lr_90 = modelo_lr_90.predict(X_test_90)

# Lasso (L1)
y_previsto_test_lasso_90 = modelo_lasso_90.predict(X_test_90)

# Ridge (L2)
y_previsto_test_ridge_90 = modelo_ridge_90.predict(X_test_90)

# ElasticNet (L1 + L2)
y_previsto_test_elasticnet_90 = modelo_elasticnet_90.predict(X_test_90)

# Decision Tree Regressor
y_previsto_test_dt_90 = modelo_dt_90.predict(X_test_90)

# Random Forest Regressor
y_previsto_test_rf_90 = modelo_rf_90.predict(X_test_90)

#-----#
# Calculando métricas #

```

```
#-----#
```

```
#-----#
```

```
# Linear Regression #
```

```
#-----#
```

```
print('-----')
```

```
print('--    LINEAR REGRESSION    --')
```

```
print('-----')
```

```
modelo_mae_test_lr_90 = mean_absolute_error(y_test_90, y_previsto_test_lr_90)
```

```
print('MAE Teste = {0:.2f}'.format(modelo_mae_test_lr_90))
```

```
modelo_mse_test_lr_90 = mean_squared_error(y_test_90, y_previsto_test_lr_90)
```

```
modelo_rmse_test_lr_90 = math.sqrt(modelo_mse_test_lr_90)
```

```
print('RMSE Teste = {0:.2f}'.format(modelo_rmse_test_lr_90))
```

```
modelo_r2_test_lr_90 = modelo_lr_90.score(X_test_90, y_test_90)
```

```
print('R² Teste = {0:.2f}'.format(modelo_r2_test_lr_90))
```

```
#-----#
```

```
# Ajuste Lasso (L1) #
```

```
#-----#
```

```
print('-----')
```

```
print('--    AJUSTE LASSO (L1)    --')
```

```
print('-----')
```

```
modelo_mae_test_lasso_90 = mean_absolute_error(y_test_90, y_previsto_test_lasso_90)
```

```
print('MAE Teste = {0:.2f}'.format(modelo_mae_test_lasso_90))
```

```
modelo_mse_test_lasso_90 = mean_squared_error(y_test_90, y_previsto_test_lasso_90)
```

```
modelo_rmse_test_lasso_90 = math.sqrt(modelo_mse_test_lasso_90)
```

```
print('RMSE Teste = {0:.2f}'.format(modelo_rmse_test_lasso_90))
```

```

modelo_r2_test_lasso_90 = modelo_lasso_90.score(X_test_90, y_test_90)
print('R² Teste = {0:.2f}'.format(modelo_r2_test_lasso_90))

```

```

#-----#
# Ajuste Ridge (L2) #
#-----#
print('-----')
print('-- AJUSTE RIDGE (L2) --')
print('-----')

```

```

modelo_mae_test_ridge_90 = mean_absolute_error(y_test_90, y_previsto_test_ridge_90)
print('MAE Teste = {0:.2f}'.format(modelo_mae_test_ridge_90))

```

```

modelo_mse_test_ridge_90 = mean_squared_error(y_test_90, y_previsto_test_ridge_90)
modelo_rmse_test_ridge_90 = math.sqrt(modelo_mse_test_ridge_90)
print('RMSE Teste = {0:.2f}'.format(modelo_rmse_test_ridge_90))

```

```

modelo_r2_test_ridge_90 = modelo_ridge_90.score(X_test_90, y_test_90)
print('R² Teste = {0:.2f}'.format(modelo_r2_test_ridge_90))

```

```

#-----#
# Ajuste ElasticNet (L1 + L2) #
#-----#
print('-----')
print('-- AJUSTE ELASTICNET (L1 + L2) --')
print('-----')

```

```

modelo_mae_test_elasticnet_90 = mean_absolute_error(y_test_90,
y_previsto_test_elasticnet_90)
print('MAE Teste = {0:.2f}'.format(modelo_mae_test_elasticnet_90))

```

```

modelo_mse_test_elasticnet_90 = mean_squared_error(y_test_90,
y_previsto_test_elasticnet_90)

```

```

modelo_rmse_test_elasticnet_90 = math.sqrt(modelo_mse_test_elasticnet_90)

```

```

print('RMSE Teste = {0:.2f}'.format(modelo_rmse_test_elasticnet_90))

```

```

modelo_r2_test_elasticnet_90 = modelo_elasticnet_90.score(X_test_90, y_test_90)

```

```

print('R² Teste = {0:.2f}'.format(modelo_r2_test_elasticnet_90))

```

```

#-----#

```

```

# Decision Tree Regressor #

```

```

#-----#

```

```

print('-----')

```

```

print('-- DECISION TREE REGRESSOR --')

```

```

print('-----')

```

```

modelo_mae_test_dt_90 = mean_absolute_error(y_test_90, y_previsto_test_dt_90)

```

```

print('MAE Teste = {0:.2f}'.format(modelo_mae_test_dt_90))

```

```

modelo_mse_test_dt_90 = mean_squared_error(y_test_90, y_previsto_test_dt_90)

```

```

modelo_rmse_test_dt_90 = math.sqrt(modelo_mse_test_dt_90)

```

```

print('RMSE Teste = {0:.2f}'.format(modelo_rmse_test_dt_90))

```

```

modelo_r2_test_dt_90 = modelo_dt_90.score(X_test_90, y_test_90)

```

```

print('R² Teste = {0:.2f}'.format(modelo_r2_test_dt_90))

```

```

#-----#

```

```

# Random Forest Regressor #

```

```

#-----#

```

```

print('-----')

```

```

print('-- RANDOM FOREST REGRESSOR --')

```

```

print('-----')

```

```
modelo_mae_test_rf_90 = mean_absolute_error(y_test_90, y_previsto_test_rf_90)
print('MAE Teste = {0:.2f}'.format(modelo_mae_test_rf_90))
```

```
modelo_mse_test_rf_90 = mean_squared_error(y_test_90, y_previsto_test_rf_90)
modelo_rmse_test_rf_90 = math.sqrt(modelo_mse_test_rf_90)
print('RMSE Teste = {0:.2f}'.format(modelo_rmse_test_rf_90))
```

```
modelo_r2_test_rf_90 = modelo_rf_90.score(X_test_90, y_test_90)
print('R² Teste = {0:.2f}'.format(modelo_r2_test_rf_90))
```

```
#-----#
```

```
# Média Aritmética para identificar potenciais pagamentos #
```

```
# Quanto menor o tempo no grupo, maior a chance de recuperação de crédito #
```

```
# Se os dias de atraso forem menores do que a média o valor do campo Recuperar será igual a 1 #
```

```
# Se os dias de atraso forem maiores ou iguais a média o valor do campo será igual a 0 #
```

```
#-----#
```

```
coluna_unica = grupo_180['Atraso'].drop_duplicates()
```

```
media_aritmetica = coluna_unica.mean()
```

```
media_aritmetica
```

```
grupo_180['Recuperar'] = grupo_180['Atraso'].apply(lambda x: 0 if x > media_aritmetica else
(1 if x < media_aritmetica else 0))
```

```
# Separar as variáveis independentes (X) e a variável dependente (y)
```

```
#-----#
```

```
# VARIÁVEIS INDEPENDENTES #
```

```
#-----#
```

```

X = grupo_180[['Saldo_Recuperado', 'Saldo', 'Atraso']]

#-----#
# VARIÁVEL DEPENDENTE #
#-----#
y = grupo_180['Recuperar']

#-----#
# DIVIDINDO DADOS EM CONJUNTOS DE TREINO E TESTE #
#-----#

# 70% PARA TREINO
# 30% PARA TESTE
X_train_180, X_test_180, y_train_180, y_test_180 = train_test_split(X, y, test_size=0.3,
random_state=42)

# Inicializar o modelo de regressão logística
modelo = LogisticRegression()

# Treinar o modelo
modelo.fit(X_train_180, y_train_180)

# Fazer previsões no conjunto de teste
previsoes = modelo.predict(X_test_180)

# Avaliar a precisão do modelo
precisao = accuracy_score(y_test_180, previsoes)
print(f'Precisão do Modelo: {precisao}')

```

```

#-----#
# INSTANCIANDO MODELOS #
#-----#

# Linear Regression
modelo_lr_180 = LinearRegression()

# Lasso (L1)
modelo_lasso_180 = Lasso()

# Ridge (L2)
modelo_ridge_180 = Ridge()

# ElasticNet (L1 + L2)
modelo_elasticnet_180 = ElasticNet()

# Decision Tree Regressor
modelo_dt_180 = DecisionTreeRegressor()

# Random Forest Regressor
modelo_rf_180 = RandomForestRegressor()

#-----#
# AJUSTANDO MODELOS #
#-----#

# Treinar o modelo
modelo_lr_180.fit(X_train_180, y_train_180)

# Lasso (L1)
modelo_lasso_180 = Lasso()

```



```

# Ridge (L2)
modelo_ridge_180 = Ridge()

# ElasticNet (L1 + L2)
modelo_elasticnet_180 = ElasticNet()

# Decision Tree Regressor
modelo_dt_180 = DecisionTreeRegressor()

# Random Forest Regressor
modelo_rf_180 = RandomForestRegressor()

#-----#
# AJUSTANDO MODELOS #
#-----#

# Linear Regression
modelo_lr_180.fit(X_train_180, y_train_180)

# Lasso (L1)
modelo_lasso_180.fit(X_train_180, y_train_180)

# Ridge (L2)
modelo_ridge_180.fit(X_train_180, y_train_180)

# ElasticNet (L1 + L2)
modelo_elasticnet_180.fit(X_train_180, y_train_180)

# Decision Tree Regressor
modelo_dt_180.fit(X_train_180, y_train_180)

# Random Forest Regressor

```

```
modelo_rf_180.fit(X_train_180, y_train_180)
```

```
#-----#
```

```
# MÉTRICAS PARA DADOS DE TREINO #
```

```
#-----#
```

```
#-----#
```

```
# Definindo um valor previsto #
```

```
#-----#
```

```
# Linear Regression
```

```
y_previsto_train_lr_180 = modelo_lr_180.predict(X_train_180)
```

```
# Lasso (L1)
```

```
y_previsto_train_lasso_180 = modelo_lasso_180.predict(X_train_180)
```

```
# Ridge (L2)
```

```
y_previsto_train_ridge_180 = modelo_ridge_180.predict(X_train_180)
```

```
# ElasticNet (L1 + L2)
```

```
y_previsto_train_elasticnet_180 = modelo_elasticnet_180.predict(X_train_180)
```

```
# Decision Tree Regressor
```

```
y_previsto_train_dt_180 = modelo_dt_180.predict(X_train_180)
```

```
# Random Forest Regressor
```

```
y_previsto_train_rf_180 = modelo_rf_180.predict(X_train_180)
```

```
#-----#
```

```
# Calculando métricas #
```

```
#-----#
```

```

#-----#
# Linear Regression #
#-----#

print('-----')
print('--    LINEAR REGRESSION    --')
print('-----')

modelo_mae_train_lr_180 = mean_absolute_error(y_train_180, y_previsto_train_lr_180)
print('MAE Treino = {0:.2f}'.format(modelo_mae_train_lr_180))

modelo_mse_train_lr_180 = mean_squared_error(y_train_180, y_previsto_train_lr_180)
modelo_rmse_train_lr_180 = math.sqrt(modelo_mse_train_lr_180)
print('RMSE Treino = {0:.2f}'.format(modelo_rmse_train_lr_180))

modelo_r2_train_lr_180 = modelo_lr_180.score(X_train_180, y_train_180)
print('R² Treino = {0:.2f}'.format(modelo_r2_train_lr_180))

#-----#
# Ajuste Lasso (L1) #
#-----#

print('-----')
print('--    AJUSTE LASSO (L1)    --')
print('-----')

modelo_mae_train_lasso_180 = mean_absolute_error(y_train_180,
y_previsto_train_lasso_180)
print('MAE Treino = {0:.2f}'.format(modelo_mae_train_lasso_180))

modelo_mse_train_lasso_180 = mean_squared_error(y_train_180,
y_previsto_train_lasso_180)
modelo_rmse_train_lasso_180 = math.sqrt(modelo_mse_train_lasso_180)
print('RMSE Treino = {0:.2f}'.format(modelo_rmse_train_lasso_180))

```

```

modelo_r2_train_lasso_180 = modelo_lasso_180.score(X_train_180, y_train_180)
print('R2 Treino = {0:.2f}'.format(modelo_r2_train_lasso_180))

```

```

#-----#
# Ajuste Ridge (L2) #
#-----#
print('-----')
print('-- AJUSTE RIDGE (L2) --')
print('-----')

```

```

modelo_mae_train_ridge_180 = mean_absolute_error(y_train_180,
y_previsto_train_ridge_180)
print('MAE Treino = {0:.2f}'.format(modelo_mae_train_ridge_180))

```

```

modelo_mse_train_ridge_180 = mean_squared_error(y_train_180,
y_previsto_train_ridge_180)
modelo_rmse_train_ridge_180 = math.sqrt(modelo_mse_train_ridge_180)
print('RMSE Treino = {0:.2f}'.format(modelo_rmse_train_ridge_180))

```

```

modelo_r2_train_ridge_180 = modelo_ridge_180.score(X_train_180, y_train_180)
print('R2 Treino = {0:.2f}'.format(modelo_r2_train_ridge_180))

```

```

#-----#
# Ajuste ElasticNet (L1 + L2) #
#-----#
print('-----')
print('-- AJUSTE ELASTICNET (L1 + L2) --')
print('-----')

```

```

modelo_mae_train_elasticnet_180 = mean_absolute_error(y_train_180,
y_previsto_train_elasticnet_180)

```

```
print('MAE Treino = {0:.2f}'.format(modelo_mae_train_elasticnet_180))
```

```
modelo_mse_train_elasticnet_180 = mean_squared_error(y_train_180,
y_previsto_train_elasticnet_180)
```

```
modelo_rmse_train_elasticnet_180 = math.sqrt(modelo_mse_train_elasticnet_180)
```

```
print('RMSE Treino = {0:.2f}'.format(modelo_rmse_train_elasticnet_180))
```

```
modelo_r2_train_elasticnet_180 = modelo_elasticnet_180.score(X_train_180, y_train_180)
```

```
print('R2 Treino = {0:.2f}'.format(modelo_r2_train_elasticnet_180))
```

```
#-----#
```

```
# Decision Tree Regressor #
```

```
#-----#
```

```
print('-----')
```

```
print('-- DECISION TREE REGRESSOR --')
```

```
print('-----')
```

```
modelo_mae_train_dt_180 = mean_absolute_error(y_train_180, y_previsto_train_dt_180)
```

```
print('MAE Treino = {0:.2f}'.format(modelo_mae_train_dt_180))
```

```
modelo_mse_train_dt_180 = mean_squared_error(y_train_180, y_previsto_train_dt_180)
```

```
modelo_rmse_train_dt_180 = math.sqrt(modelo_mse_train_dt_180)
```

```
print('RMSE Treino = {0:.2f}'.format(modelo_rmse_train_dt_180))
```

```
modelo_r2_train_dt_180 = modelo_dt_180.score(X_train_180, y_train_180)
```

```
print('R2 Treino = {0:.2f}'.format(modelo_r2_train_dt_180))
```

```
#-----#
```

```
# Random Forest Regressor #
```

```
#-----#
```

```
print('-----')
```

```
print('-- RANDOM FOREST REGRESSOR --')
```

```

print('-----')

modelo_mae_train_rf_180 = mean_absolute_error(y_train_180, y_previsto_train_rf_180)
print('MAE Treino = {0:.2f}'.format(modelo_mae_train_rf_180))

modelo_mse_train_rf_180 = mean_squared_error(y_train_180, y_previsto_train_rf_180)
modelo_rmse_train_rf_180 = math.sqrt(modelo_mse_train_rf_180)
print('RMSE Treino = {0:.2f}'.format(modelo_rmse_train_rf_180))

modelo_r2_train_rf_180 = modelo_rf_180.score(X_train_180, y_train_180)
print('R² Treino = {0:.2f}'.format(modelo_r2_train_rf_180))

#-----#
# MÉTRICAS PARA DADOS DE TESTE #
#-----#

#-----#
# Definindo um valor previsto #
#-----#

# Linear Regression
y_previsto_test_lr_180 = modelo_lr_180.predict(X_test_180)

# Lasso (L1)
y_previsto_test_lasso_180 = modelo_lasso_180.predict(X_test_180)

# Ridge (L2)
y_previsto_test_ridge_180 = modelo_ridge_180.predict(X_test_180)

# ElasticNet (L1 + L2)
y_previsto_test_elasticnet_180 = modelo_elasticnet_180.predict(X_test_180)

```

```

# Decision Tree Regressor
y_previsto_test_dt_180 = modelo_dt_180.predict(X_test_180)

# Random Forest Regressor
y_previsto_test_rf_180 = modelo_rf_180.predict(X_test_180)

#-----#
# Calculando métricas #
#-----#

#-----#
# Linear Regression #
#-----#
print('-----')
print('--    LINEAR REGRESSION    --')
print('-----')

modelo_mae_test_lr_180 = mean_absolute_error(y_test_180, y_previsto_test_lr_180)
print('MAE Teste = {0:.2f}'.format(modelo_mae_test_lr_180))

modelo_mse_test_lr_180 = mean_squared_error(y_test_180, y_previsto_test_lr_180)
modelo_rmse_test_lr_180 = math.sqrt(modelo_mse_test_lr_180)
print('RMSE Teste = {0:.2f}'.format(modelo_rmse_test_lr_180))

modelo_r2_test_lr_180 = modelo_lr_180.score(X_test_180, y_test_180)
print('R² Teste = {0:.2f}'.format(modelo_r2_test_lr_180))

#-----#
# Ajuste Lasso (L1) #
#-----#
print('-----')
print('--    AJUSTE LASSO (L1)    --')

```

```
print('-----')
```

```
modelo_mae_test_lasso_180 = mean_absolute_error(y_test_180,
y_previsto_test_lasso_180)
print('MAE Teste = {0:.2f}'.format(modelo_mae_test_lasso_30))
```

```
modelo_mse_test_lasso_180 = mean_squared_error(y_test_180,
y_previsto_test_lasso_180)
modelo_rmse_test_lasso_180 = math.sqrt(modelo_mse_test_lasso_30)
print('RMSE Teste = {0:.2f}'.format(modelo_rmse_test_lasso_30))
```

```
modelo_r2_test_lasso_180 = modelo_lasso_180.score(X_test_180, y_test_180)
print('R² Teste = {0:.2f}'.format(modelo_r2_test_lasso_30))
```

```
#-----#
```

```
# Ajuste Ridge (L2) #
```

```
#-----#
```

```
print('-----')
print('-- AJUSTE RIDGE (L2) --')
print('-----')
```

```
modelo_mae_test_ridge_180 = mean_absolute_error(y_test_180,
y_previsto_test_ridge_180)
print('MAE Teste = {0:.2f}'.format(modelo_mae_test_ridge_180))
```

```
modelo_mse_test_ridge_180 = mean_squared_error(y_test_180,
y_previsto_test_ridge_180)
modelo_rmse_test_ridge_180 = math.sqrt(modelo_mse_test_ridge_180)
print('RMSE Teste = {0:.2f}'.format(modelo_rmse_test_ridge_180))
```

```
modelo_r2_test_ridge_180 = modelo_ridge_180.score(X_test_180, y_test_180)
print('R² Teste = {0:.2f}'.format(modelo_r2_test_ridge_180))
```



```

#-----#
# Ajuste ElastciNet (L1 + L2) #
#-----#
print('-----')
print('-- AJUSTE ELASTICNET (L1 + L2) --')
print('-----')

modelo_mae_test_elasticnet_180 = mean_absolute_error(y_test_180,
y_previsto_test_elasticnet_180)
print('MAE Teste = {0:.2f}'.format(modelo_mae_test_elasticnet_180))

modelo_mse_test_elasticnet_180 = mean_squared_error(y_test_180,
y_previsto_test_elasticnet_180)
modelo_rmse_test_elasticnet_180 = math.sqrt(modelo_mse_test_elasticnet_180)
print('RMSE Teste = {0:.2f}'.format(modelo_rmse_test_elasticnet_180))

modelo_r2_test_elasticnet_180 = modelo_elasticnet_180.score(X_test_180, y_test_180)
print('R² Teste = {0:.2f}'.format(modelo_r2_test_elasticnet_180))

#-----#
# Decision Tree Regressor #
#-----#
print('-----')
print('-- DECISION TREE REGRESSOR --')
print('-----')

modelo_mae_test_dt_180 = mean_absolute_error(y_test_180, y_previsto_test_dt_180)
print('MAE Teste = {0:.2f}'.format(modelo_mae_test_dt_180))

modelo_mse_test_dt_180 = mean_squared_error(y_test_180, y_previsto_test_dt_180)
modelo_rmse_test_dt_180 = math.sqrt(modelo_mse_test_dt_180)

```

```
print('RMSE Teste = {0:.2f}'.format(modelo_rmse_test_dt_180))
```

```
modelo_r2_test_dt_180 = modelo_dt_180.score(X_test_180, y_test_180)
```

```
print('R² Teste = {0:.2f}'.format(modelo_r2_test_dt_180))
```

```
#-----#
```

```
# Random Forest Regressor #
```

```
#-----#
```

```
print('-----')
```

```
print('-- RANDOM FOREST REGRESSOR --')
```

```
print('-----')
```

```
modelo_mae_test_rf_180 = mean_absolute_error(y_test_180, y_previsto_test_rf_180)
```

```
print('MAE Teste = {0:.2f}'.format(modelo_mae_test_rf_180))
```

```
modelo_mse_test_rf_180 = mean_squared_error(y_test_180, y_previsto_test_rf_180)
```

```
modelo_rmse_test_rf_180 = math.sqrt(modelo_mse_test_rf_180)
```

```
print('RMSE Teste = {0:.2f}'.format(modelo_rmse_test_rf_180))
```

```
modelo_r2_test_rf_180 = modelo_rf_180.score(X_test_180, y_test_180)
```

```
print('R² Teste = {0:.2f}'.format(modelo_r2_test_rf_180))
```

```
#-----#
```

```
# Média Aritmética para identificar potenciais pagamentos #
```

```
# Quanto menor o tempo no grupo, maior a chance de recuperação de crédito #
```

```
# Se os dias de atraso forem menores do que a média o valor do campo Recuperar será igual a 1 #
```

```
# Se os dias de atraso forem maiores ou iguais a média o valor do campo será igual a 0 #
```

```
#-----#
```

```
coluna_unica = grupo_365['Atraso'].drop_duplicates()
```

```
media_aritmetica = coluna_unica.mean()
```

```
media_aritmetica
```

```
grupo_365['Recuperar'] = grupo_365['Atraso'].apply(lambda x: 0 if x > media_aritmetica else
(1 if x < media_aritmetica else 0))
```

```
# Separar as variáveis independentes (X) e a variável dependente (y)
```

```
#-----#
```

```
# VARIÁVEIS INDEPENDENTES #
```

```
#-----#
```

```
X = grupo_365[['Saldo_Recuperado', 'Saldo', 'Atraso']]
```

```
#-----#
```

```
# VARIÁVEL DEPENDENTE #
```

```
#-----#
```

```
y = grupo_365['Recuperar']
```

```
#-----#
```

```
# DIVIDINDO DADOS EM CONJUNTOS DE TREINO E TESTE #
```

```
#-----#
```

```
# 70% PARA TREINO
```

```
# 30% PARA TESTE
```

```
X_train_365, X_test_365, y_train_365, y_test_365 = train_test_split(X, y, test_size=0.3,
random_state=42)
```

```
# Inicializar o modelo de regressão logística
```

```
modelo = LogisticRegression()
```

```
# Treinar o modelo
```

```
modelo.fit(X_train_365, y_train_365)

# Fazer previsões no conjunto de teste
previsoes = modelo.predict(X_test_365)

# Avaliar a precisão do modelo
precisao = accuracy_score(y_test_365, previsoes)
print(f'Precisão do Modelo: {precisao}')

#-----#
# INSTANCIANDO MODELOS #
#-----#

# Linear Regression
modelo_lr_365 = LinearRegression()

# Lasso (L1)
modelo_lasso_365 = Lasso()

# Ridge (L2)
modelo_ridge_365 = Ridge()

# ElasticNet (L1 + L2)
modelo_elasticnet_365 = ElasticNet()

# Decision Tree Regressor
modelo_dt_365 = DecisionTreeRegressor()

# Random Forest Regressor
modelo_rf_365 = RandomForestRegressor()
```

```

#-----#
# AJUSTANDO MODELOS #
#-----#

# Treinar o modelo
modelo_lr_365.fit(X_train_365, y_train_365)

# Lasso (L1)
modelo_lasso_365 = Lasso()

# Ridge (L2)
modelo_ridge_365 = Ridge()

# ElasticNet (L1 + L2)
modelo_elasticnet_365 = ElasticNet()

# Decision Tree Regressor
modelo_dt_365 = DecisionTreeRegressor()

# Random Forest Regressor
modelo_rf_365 = RandomForestRegressor()

#-----#
# AJUSTANDO MODELOS #
#-----#

# Linear Regression
modelo_lr_365.fit(X_train_365, y_train_365)

# Lasso (L1)
modelo_lasso_365.fit(X_train_365, y_train_365)

```

Ridge (L2)

modelo_ridge_365.fit(X_train_365, y_train_365)

ElasticNet (L1 + L2)

modelo_elasticnet_365.fit(X_train_365, y_train_365)

Decision Tree Regressor

modelo_dt_365.fit(X_train_365, y_train_365)

Random Forest Regressor

modelo_rf_365.fit(X_train_365, y_train_365)

#-----#

MÉTRICAS PARA DADOS DE TREINO

#-----#

#-----#

Definindo um valor previsto

#-----#

Linear Regression

y_previsto_train_lr_365 = modelo_lr_365.predict(X_train_365)

Lasso (L1)

y_previsto_train_lasso_365 = modelo_lasso_365.predict(X_train_365)

Ridge (L2)

y_previsto_train_ridge_365 = modelo_ridge_365.predict(X_train_365)

ElasticNet (L1 + L2)

y_previsto_train_elasticnet_365 = modelo_elasticnet_365.predict(X_train_365)

```

# Decision Tree Regressor
y_previsto_train_dt_365 = modelo_dt_365.predict(X_train_365)

# Random Forest Regressor
y_previsto_train_rf_365 = modelo_rf_365.predict(X_train_365)

#-----#
# Calculando métricas #
#-----#

#-----#
# Linear Regression #
#-----#
print('-----')
print('--    LINEAR REGRESSION    --')
print('-----')

modelo_mae_train_lr_365 = mean_absolute_error(y_train_365, y_previsto_train_lr_365)
print('MAE Treino = {0:.2f}'.format(modelo_mae_train_lr_365))

modelo_mse_train_lr_365 = mean_squared_error(y_train_365, y_previsto_train_lr_365)
modelo_rmse_train_lr_365 = math.sqrt(modelo_mse_train_lr_365)
print('RMSE Treino = {0:.2f}'.format(modelo_rmse_train_lr_365))

modelo_r2_train_lr_365 = modelo_lr_365.score(X_train_30, y_train_30)
print('R² Treino = {0:.2f}'.format(modelo_r2_train_lr_365))

#-----#
# Ajuste Lasso (L1) #
#-----#
print('-----')
print('--    AJUSTE LASSO (L1)    --')

```

```
print('-----')
```

```
modelo_mae_train_lasso_365 = mean_absolute_error(y_train_365,
y_previsto_train_lasso_365)
print('MAE Treino = {0:.2f}'.format(modelo_mae_train_lasso_365))
```

```
modelo_mse_train_lasso_365 = mean_squared_error(y_train_365,
y_previsto_train_lasso_365)
modelo_rmse_train_lasso_365 = math.sqrt(modelo_mse_train_lasso_365)
print('RMSE Treino = {0:.2f}'.format(modelo_rmse_train_lasso_365))
```

```
modelo_r2_train_lasso_365 = modelo_lasso_365.score(X_train_30, y_train_30)
print('R² Treino = {0:.2f}'.format(modelo_r2_train_lasso_365))
```

```
#-----#
```

```
# Ajuste Ridge (L2) #
```

```
#-----#
```

```
print('-----')
print('-- AJUSTE RIDGE (L2) --')
print('-----')
```

```
modelo_mae_train_ridge_365 = mean_absolute_error(y_train_365,
y_previsto_train_ridge_365)
print('MAE Treino = {0:.2f}'.format(modelo_mae_train_ridge_365))
```

```
modelo_mse_train_ridge_365 = mean_squared_error(y_train_365,
y_previsto_train_ridge_365)
modelo_rmse_train_ridge_365 = math.sqrt(modelo_mse_train_ridge_365)
print('RMSE Treino = {0:.2f}'.format(modelo_rmse_train_ridge_365))
```

```
modelo_r2_train_ridge_365 = modelo_ridge_365.score(X_train_365, y_train_365)
print('R² Treino = {0:.2f}'.format(modelo_r2_train_ridge_365))
```



```

#-----#
# Ajuste ElasticNet (L1 + L2) #
#-----#
print('-----')
print('-- AJUSTE ELASTICNET (L1 + L2) --')
print('-----')

modelo_mae_train_elasticnet_365 = mean_absolute_error(y_train_365,
y_previsto_train_elasticnet_365)
print('MAE Treino = {0:.2f}'.format(modelo_mae_train_elasticnet_365))

modelo_mse_train_elasticnet_365 = mean_squared_error(y_train_365,
y_previsto_train_elasticnet_365)
modelo_rmse_train_elasticnet_365 = math.sqrt(modelo_mse_train_elasticnet_365)
print('RMSE Treino = {0:.2f}'.format(modelo_rmse_train_elasticnet_365))

modelo_r2_train_elasticnet_365 = modelo_elasticnet_365.score(X_train_365, y_train_365)
print('R² Treino = {0:.2f}'.format(modelo_r2_train_elasticnet_365))

#-----#
# Decision Tree Regressor #
#-----#
print('-----')
print('-- DECISION TREE REGRESSOR --')
print('-----')

modelo_mae_train_dt_365 = mean_absolute_error(y_train_365, y_previsto_train_dt_365)
print('MAE Treino = {0:.2f}'.format(modelo_mae_train_dt_365))

modelo_mse_train_dt_365 = mean_squared_error(y_train_365, y_previsto_train_dt_365)
modelo_rmse_train_dt_365 = math.sqrt(modelo_mse_train_dt_365)

```

```

print('RMSE Treino = {0:.2f}'.format(modelo_rmse_train_dt_365))

modelo_r2_train_dt_365 = modelo_dt_365.score(X_train_365, y_train_365)
print('R² Treino = {0:.2f}'.format(modelo_r2_train_dt_365))

#-----#
# Random Forest Regressor #
#-----#
print('-----')
print('-- RANDOM FOREST REGRESSOR --')
print('-----')

modelo_mae_train_rf_365 = mean_absolute_error(y_train_365, y_previsto_train_rf_365)
print('MAE Treino = {0:.2f}'.format(modelo_mae_train_rf_365))

modelo_mse_train_rf_365 = mean_squared_error(y_train_365, y_previsto_train_rf_365)
modelo_rmse_train_rf_365 = math.sqrt(modelo_mse_train_rf_365)
print('RMSE Treino = {0:.2f}'.format(modelo_rmse_train_rf_365))

modelo_r2_train_rf_365 = modelo_rf_365.score(X_train_365, y_train_365)
print('R² Treino = {0:.2f}'.format(modelo_r2_train_rf_365))

#-----#
# MÉTRICAS PARA DADOS DE TESTE #
#-----#

#-----#
# Definindo um valor previsto #
#-----#

# Linear Regression
y_previsto_test_lr_365 = modelo_lr_365.predict(X_test_365)

```

```
# Lasso (L1)
```

```
y_previsto_test_lasso_365 = modelo_lasso_365.predict(X_test_365)
```

```
# Ridge (L2)
```

```
y_previsto_test_ridge_365 = modelo_ridge_365.predict(X_test_365)
```

```
# ElasticNet (L1 + L2)
```

```
y_previsto_test_elasticnet_365 = modelo_elasticnet_365.predict(X_test_365)
```

```
# Decision Tree Regressor
```

```
y_previsto_test_dt_365 = modelo_dt_365.predict(X_test_365)
```

```
# Random Forest Regressor
```

```
y_previsto_test_rf_365 = modelo_rf_365.predict(X_test_365)
```

```
#-----#
```

```
# Calculando métricas #
```

```
#-----#
```

```
#-----#
```

```
# Linear Regression #
```

```
#-----#
```

```
print('-----')
```

```
print('--    LINEAR REGRESSION    --')
```

```
print('-----')
```

```
modelo_mae_test_lr_365 = mean_absolute_error(y_test_365, y_previsto_test_lr_365)
```

```
print('MAE Teste = {0:.2f}'.format(modelo_mae_test_lr_365))
```

```
modelo_mse_test_lr_365 = mean_squared_error(y_test_365, y_previsto_test_lr_365)
```

```
modelo_rmse_test_lr_365 = math.sqrt(modelo_mse_test_lr_365)
```

```
print('RMSE Teste = {0:.2f}'.format(modelo_rmse_test_lr_365))
```

```
modelo_r2_test_lr_365 = modelo_lr_365.score(X_test_365, y_test_365)
```

```
print('R² Teste = {0:.2f}'.format(modelo_r2_test_lr_365))
```

```
#-----#
```

```
# Ajuste Lasso (L1) #
```

```
#-----#
```

```
print('-----')
```

```
print('-- AJUSTE LASSO (L1) --')
```

```
print('-----')
```

```
modelo_mae_test_lasso_365 = mean_absolute_error(y_test_365,
y_previsto_test_lasso_365)
```

```
print('MAE Teste = {0:.2f}'.format(modelo_mae_test_lasso_365))
```

```
modelo_mse_test_lasso_365 = mean_squared_error(y_test_365,
y_previsto_test_lasso_365)
```

```
modelo_rmse_test_lasso_365 = math.sqrt(modelo_mse_test_lasso_365)
```

```
print('RMSE Teste = {0:.2f}'.format(modelo_rmse_test_lasso_365))
```

```
modelo_r2_test_lasso_365 = modelo_lasso_365.score(X_test_365, y_test_365)
```

```
print('R² Teste = {0:.2f}'.format(modelo_r2_test_lasso_365))
```

```
#-----#
```

```
# Ajuste Ridge (L2) #
```

```
#-----#
```

```
print('-----')
```

```
print('-- AJUSTE RIDGE (L2) --')
```

```
print('-----')
```

```

modelo_mae_test_ridge_365          =          mean_absolute_error(y_test_365,
y_previsto_test_ridge_365)
print('MAE Teste = {0:.2f}'.format(modelo_mae_test_ridge_365))

```

```

modelo_mse_test_ridge_365          =          mean_squared_error(y_test_365,
y_previsto_test_ridge_365)
modelo_rmse_test_ridge_365 = math.sqrt(modelo_mse_test_ridge_365)
print('RMSE Teste = {0:.2f}'.format(modelo_rmse_test_ridge_365))

```

```

modelo_r2_test_ridge_365 = modelo_ridge_365.score(X_test_365, y_test_365)
print('R² Teste = {0:.2f}'.format(modelo_r2_test_ridge_365))

```

```

#-----#
# Ajuste ElastciNet (L1 + L2) #
#-----#
print('-----')
print('-- AJUSTE ELASTICNET (L1 + L2) --')
print('-----')

```

```

modelo_mae_test_elasticnet_365      =          mean_absolute_error(y_test_365,
y_previsto_test_elasticnet_365)
print('MAE Teste = {0:.2f}'.format(modelo_mae_test_elasticnet_365))

```

```

modelo_mse_test_elasticnet_365      =          mean_squared_error(y_test_365,
y_previsto_test_elasticnet_365)
modelo_rmse_test_elasticnet_365 = math.sqrt(modelo_mse_test_elasticnet_365)
print('RMSE Teste = {0:.2f}'.format(modelo_rmse_test_elasticnet_365))

```

```

modelo_r2_test_elasticnet_365 = modelo_elasticnet_365.score(X_test_365, y_test_365)
print('R² Teste = {0:.2f}'.format(modelo_r2_test_elasticnet_365))

```

```

#-----#

```

```

# Decision Tree Regressor #
#-----#
print('-----')
print('-- DECISION TREE REGRESSOR --')
print('-----')

modelo_mae_test_dt_365 = mean_absolute_error(y_test_365, y_previsto_test_dt_365)
print('MAE Teste = {0:.2f}'.format(modelo_mae_test_dt_365))

modelo_mse_test_dt_365 = mean_squared_error(y_test_365, y_previsto_test_dt_365)
modelo_rmse_test_dt_365 = math.sqrt(modelo_mse_test_dt_365)
print('RMSE Teste = {0:.2f}'.format(modelo_rmse_test_dt_365))

modelo_r2_test_dt_365 = modelo_dt_365.score(X_test_365, y_test_365)
print('R² Teste = {0:.2f}'.format(modelo_r2_test_dt_365))

#-----#
# Random Forest Regressor #
#-----#
print('-----')
print('-- RANDOM FOREST REGRESSOR --')
print('-----')

modelo_mae_test_rf_365 = mean_absolute_error(y_test_365, y_previsto_test_rf_365)
print('MAE Teste = {0:.2f}'.format(modelo_mae_test_rf_365))

modelo_mse_test_rf_365 = mean_squared_error(y_test_365, y_previsto_test_rf_365)
modelo_rmse_test_rf_365 = math.sqrt(modelo_mse_test_rf_365)
print('RMSE Teste = {0:.2f}'.format(modelo_rmse_test_rf_365))

modelo_r2_test_rf_365 = modelo_rf_365.score(X_test_365, y_test_365)
print('R² Teste = {0:.2f}'.format(modelo_r2_test_rf_365))

```