Projeto 01 - Classificação de crédito

April 28, 2025

1 Projeto 01 - Concessão de cartões de crédito

Este notebook é semelhante ao visto em vídeo, mas contém células azuis como esta, que trazem instruções para a sua atividade.

Após realizar as tarefas indicadas, você vai fazer o upload do seu arquivo no GitHub e enviar o link para a EBAC, ou alternativamente, fazer o upload do arquivo na plataforma da EBAC. Recomendamos o github, pois assim você já vai montando o seu portfólio.

1.1 Etapa 1 CRISP - DM: Entendimento do negócio

Como primeira etapa do CRISP-DM, vamos entender do que se trata o negócio, e quais os objetivos.

Este é um problema de concessão de cartões de crédito, publicado no Kaggle, uma plataforma que promove desafios de ciência de dados, oferecendo prêmios em dinheiro para os melhores colocados. O link original está aqui.

Essa é uma base de proponentes de cartão de crédito, nosso objetivo é construir um modelo preditivo para identificar o risco de inadimplência (tipicamente definida pela ocorrência de um atraso maior ou igual a 90 em um horizonte de 12 meses) através de variáveis que podem ser observadas na data da avaliação do crédito (tipicamente quando o cliente solicita o cartão).

Atividades do CRISP-DM:

- Objetivos do negócio Note que o objetivo aqui é que o modelo sirva o mutuário (o cliente) para que avalie suas próprias decisões, e não a instituição de crédito.
- Objetivos da modelagem O objetivo está bem definido: desenvolver o melhor modelo preditivo de modo a auxiliar o mutuário a tomar suas próprias decisões referentes a crédito.

Nessa etapa também se avalia a situação da empresa/segmento/assunto de modo a se entender o tamanho do público, relevância, problemas presentes e todos os detalhes do processo gerador do fenômeno em questão, e portanto dos dados.

Também é nessa etapa que se constrói um planejamento do projeto.

1.2 Etapa 2 Crisp-DM: Entendimento dos dados

A segunda etapa é o entendimento dos dados. Foram fornecidas 15 variáveis mais a variável resposta (em negrito na tabela). O significado de cada uma dessas variáveis se encontra na tabela.

Dicionário de dados Os dados estão dispostos em uma tabela com uma linha para cada cliente, e uma coluna para cada variável armazenando as características desses clientes. Colocamos uma cópia o dicionário de dados (explicação dessas variáveis) abaixo neste notebook:

Variable Name	Description	Tipo
sexo	M = 'Masculino'; F = 'Feminino'	M/F
posse_de_veiculo	Y = 'possui'; N = 'não possui'	Y/N
posse_de_imovel	Y = 'possui'; N = 'não possui'	Y/N
qtd_filhos	Quantidade de filhos	inteiro
tipo_renda	Tipo de renda (ex: assaliariado, autônomo etc)	texto
educacao	Nível de educação (ex: secundário, superior etc)	texto
estado_civil	Estado civil (ex: solteiro, casado etc)	texto
tipo_residencia	tipo de residência (ex: casa/apartamento, com os pais etc)	texto
idade	idade em anos	inteiro
tempo de emprego	tempo de emprego em anos	inteiro
possui_celular	Indica se possui celular (1 = sim, 0 = não)	binária
possui_fone_comercial	Indica se possui telefone comercial (1 = sim, 0 = não)	binária
possui_fone	Indica se possui telefone (1 = sim, 0 = não)	binária
possui_email	Indica se possui e-mail (1 = sim, 0 = não)	binária
qt_pessoas_residencia	quantidade de pessoas na residência	inteiro
mau	indicadora de mau pagador (True = mau, False = bom)	binária

Carregando os pacotes É considerado uma boa prática carregar os pacotes que serão utilizados como a primeira coisa do programa.

[1]:
```
import pandas as pd

import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.ensemble import RandomForestClassifier
```

Carregando os dados O comando pd.read_csv é um comando da biblioteca pandas (pd.) e carrega os dados do arquivo csv indicado para um objeto *dataframe* do pandas.

[2]:
```
# Observe que demo01.csv está na mesma pasta que este notebook
# do contrário, seria necessário indicar a pasta no nome do arquivo
df = pd.read_csv('demo01.csv')
print ("Número de linhas e colunas da tabela: {}".format(df.shape))

df.head()
```

```
Número de linhas e colunas da tabela: (16650, 16)
```

[2]:
```
  sexo posse_de_veiculo posse_de_imovel  qtd_filhos            tipo_renda  \
0    M                Y               Y           0               Working
1    F                N               Y           0  Commercial associate
2    F                N               Y           0  Commercial associate
3    M                Y               Y           0               Working
4    F                Y               N           0               Working

                        educacao          estado_civil    tipo_residencia  \
0  Secondary / secondary special               Married  House / apartment
1  Secondary / secondary special  Single / not married  House / apartment
2  Secondary / secondary special  Single / not married  House / apartment
3               Higher education               Married  House / apartment
4              Incomplete higher               Married  House / apartment

       idade  tempo_emprego  possui_celular  possui_fone_comercial  \
0  58.832877       3.106849               1                      0
1  52.356164       8.358904               1                      0
2  52.356164       8.358904               1                      0
3  46.224658       2.106849               1                      1
4  29.230137       3.021918               1                      0

   possui_fone  possui_email  qt_pessoas_residencia    mau
0            0             0                    2.0  False
1            1             1                    1.0  False
2            1             1                    1.0  False
3            1             1                    2.0  False
4            0             0                    2.0  False
```

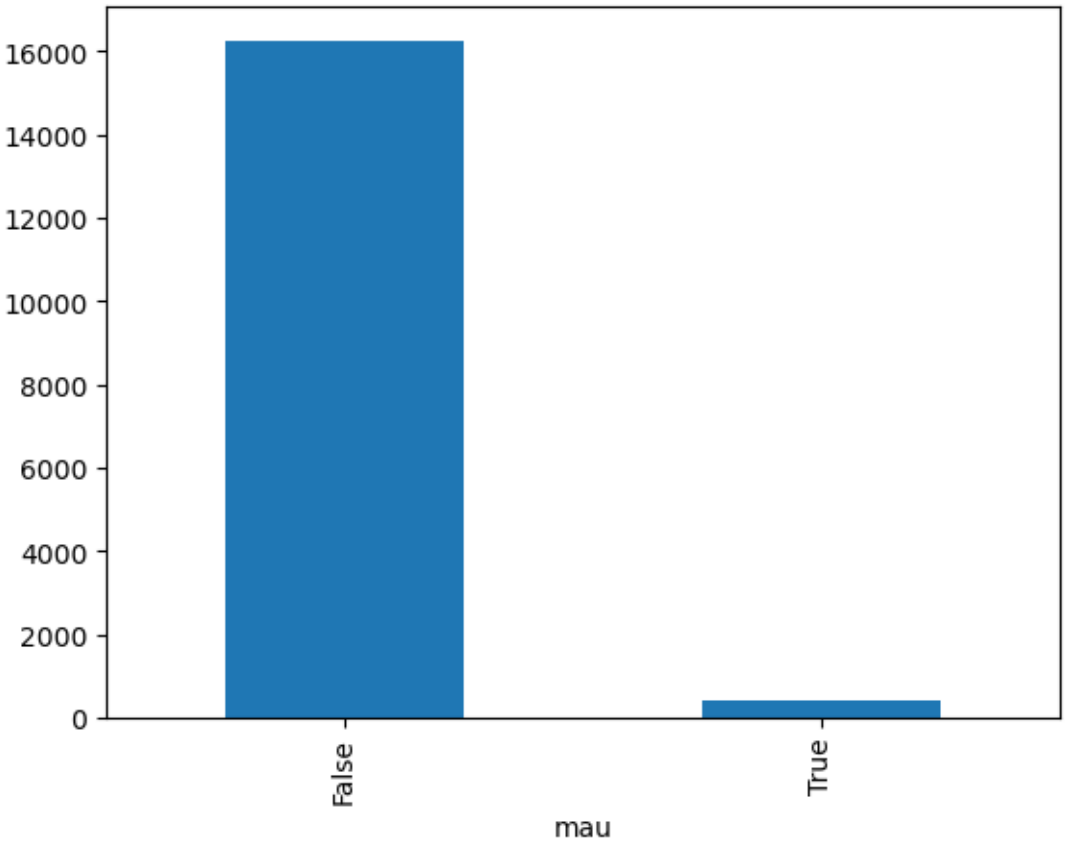
Entendimento dos dados - Univariada Nesta etapa tipicamente avaliamos a distribuição de todas as variáveis. Nesta demonstração vamos ver a variável resposta e dois exemplos de univariada apenas. Mas sinta-se à vontade para tentar observar outras variáveis.

[3]:
```
print(df['mau'].value_counts())
print("\nTaxa de inadimplentes:")
print(df['mau'].mean())
```

```
mau
False    16260
True       390
Name: count, dtype: int64

Taxa de inadimplentes:
0.023423423423423424
```

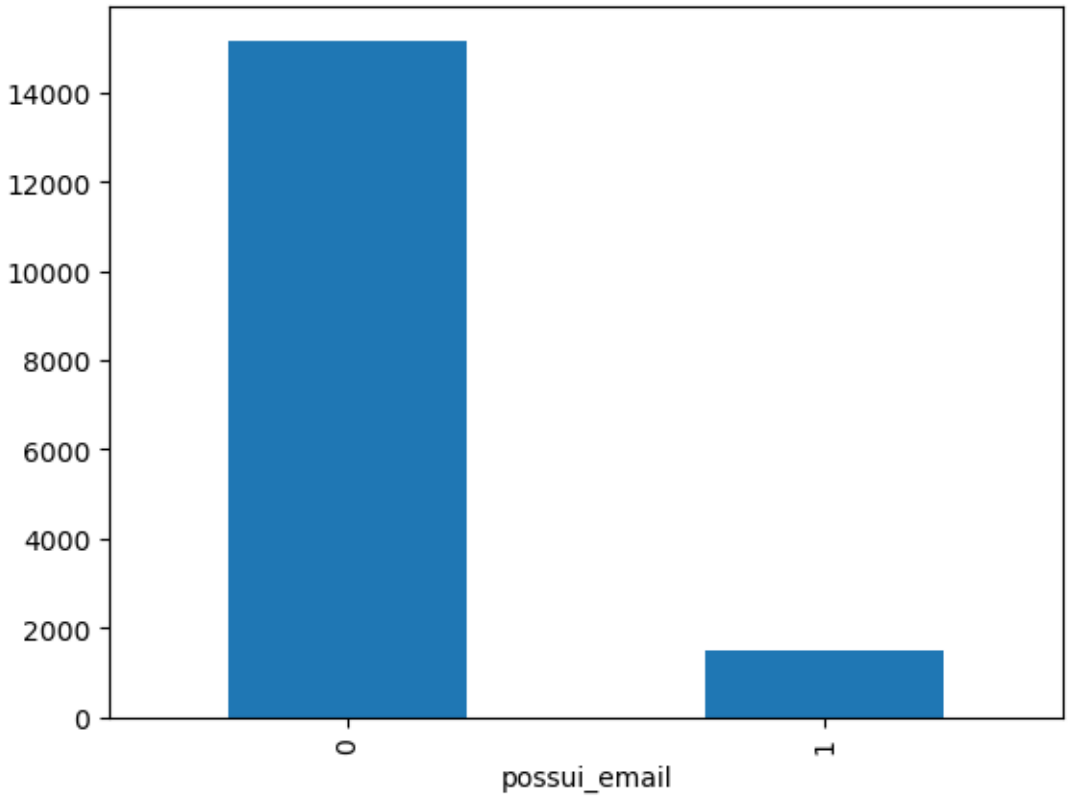
[4]:
```
var = 'mau'
grafico_barras = df[var].value_counts().plot.bar()
```



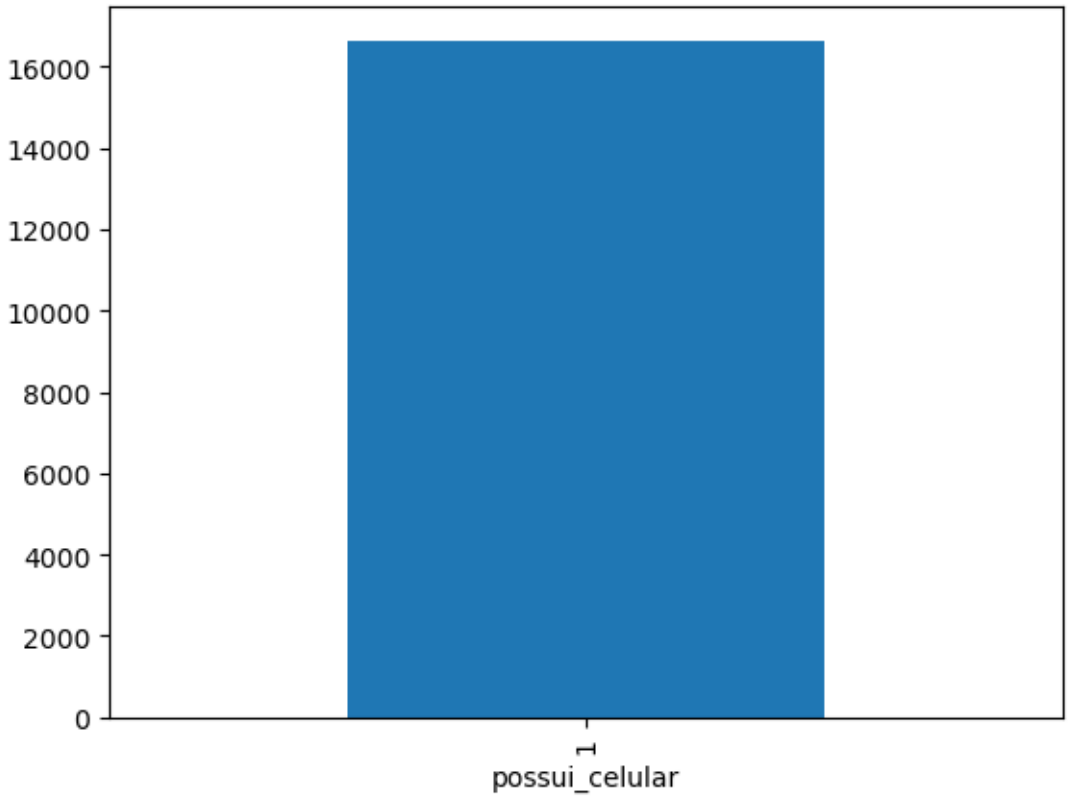
1.2.1 Tarefa 01 - gráfico de barras

Com base no código da célula anterior, construa um gráfico de barras para pelo menos duas outras variáveis. **Dica:** Não tente usar as variáveis `tempo_emprego` e `idade` pois o gráfico de barras dessa forma como construímos não é adequado para elas.

[5]:
```
var = 'possui_email'
grafico_barras = df[var].value_counts().plot.bar()
```



[6]:
```
var = 'possui_celular'
grafico_barras = df[var].value_counts().plot.bar()
```

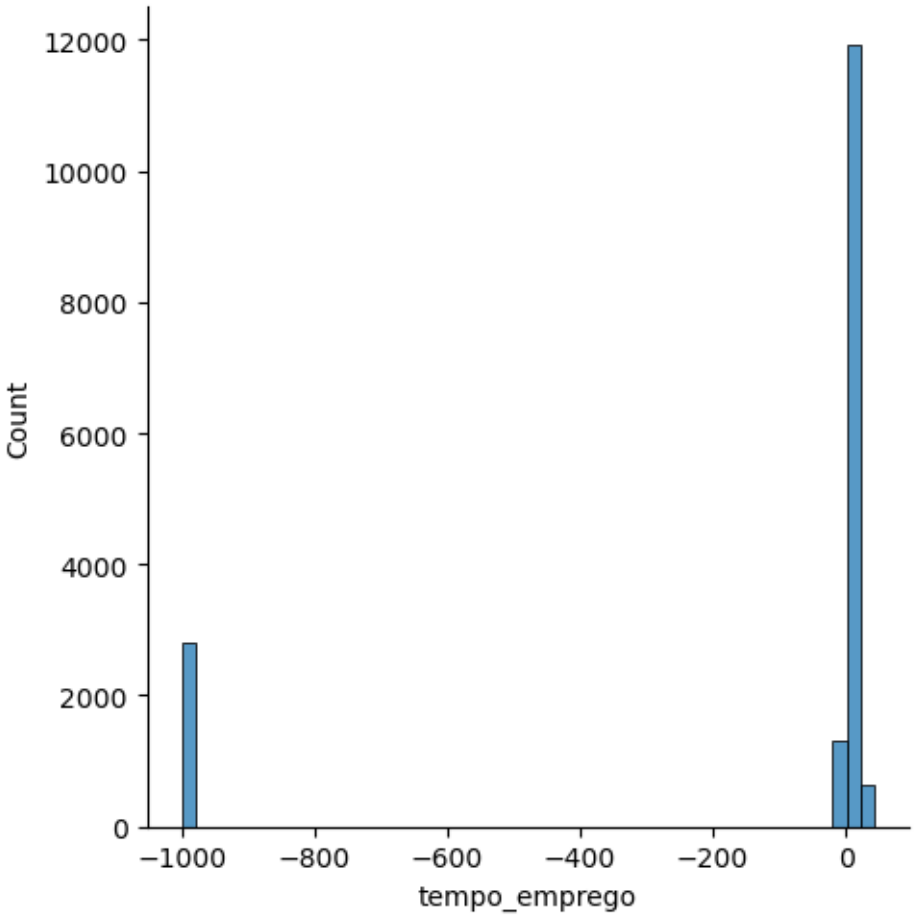


[7]:
```
plt.clf()
var = "tempo_emprego"

sns.displot(df, x = var, bins = 50)
plt.show()
```

```
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages/seaborn/axisgrid.py:118: UserWarning: The figure layout has changed to
tight
  self._figure.tight_layout(*args, **kwargs)
```

```
<Figure size 640x480 with 0 Axes>
```



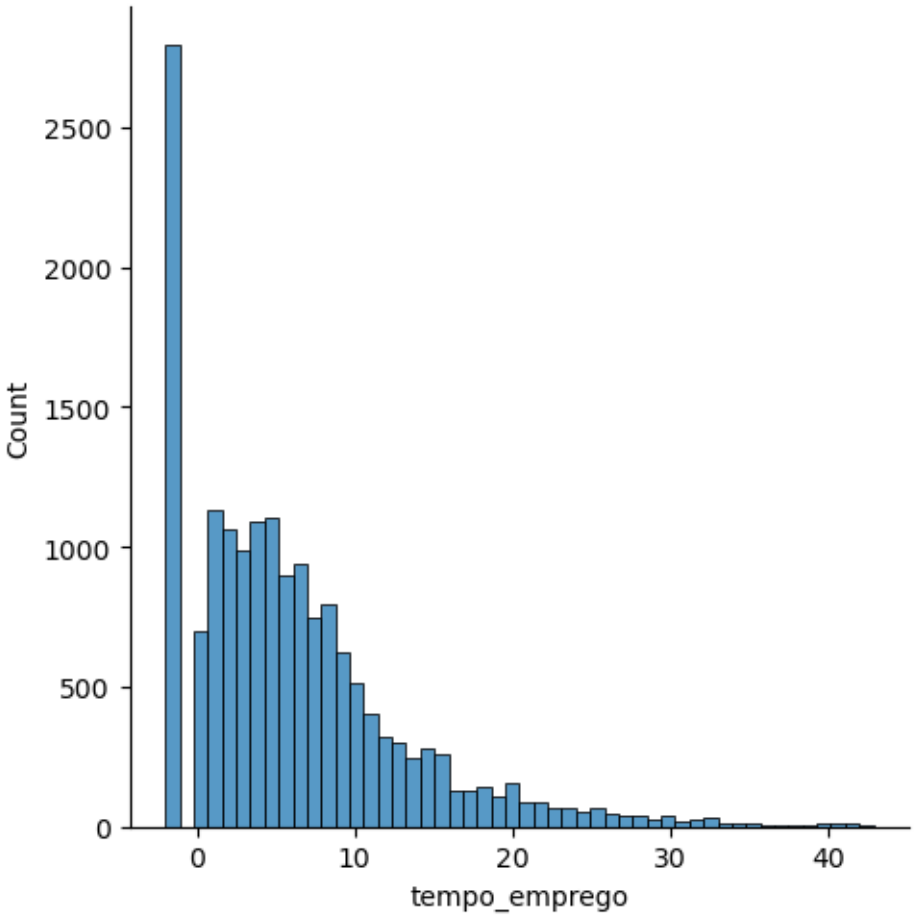
[8]:
```
# Alterando valores de -1000 pra -2, para visualizar melhor no gráfico
var = "tempo_emprego"
df.loc[df[var]<0,var] = -2
```

[9]:
```
plt.clf()
var = "tempo_emprego"

sns.displot(df, x = var, bins = 50)
plt.show()
```

```
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages/seaborn/axisgrid.py:118: UserWarning: The figure layout has changed to
tight
  self._figure.tight_layout(*args, **kwargs)
```

```
<Figure size 640x480 with 0 Axes>
```



1.2.2 Tarefa 02 - Histograma

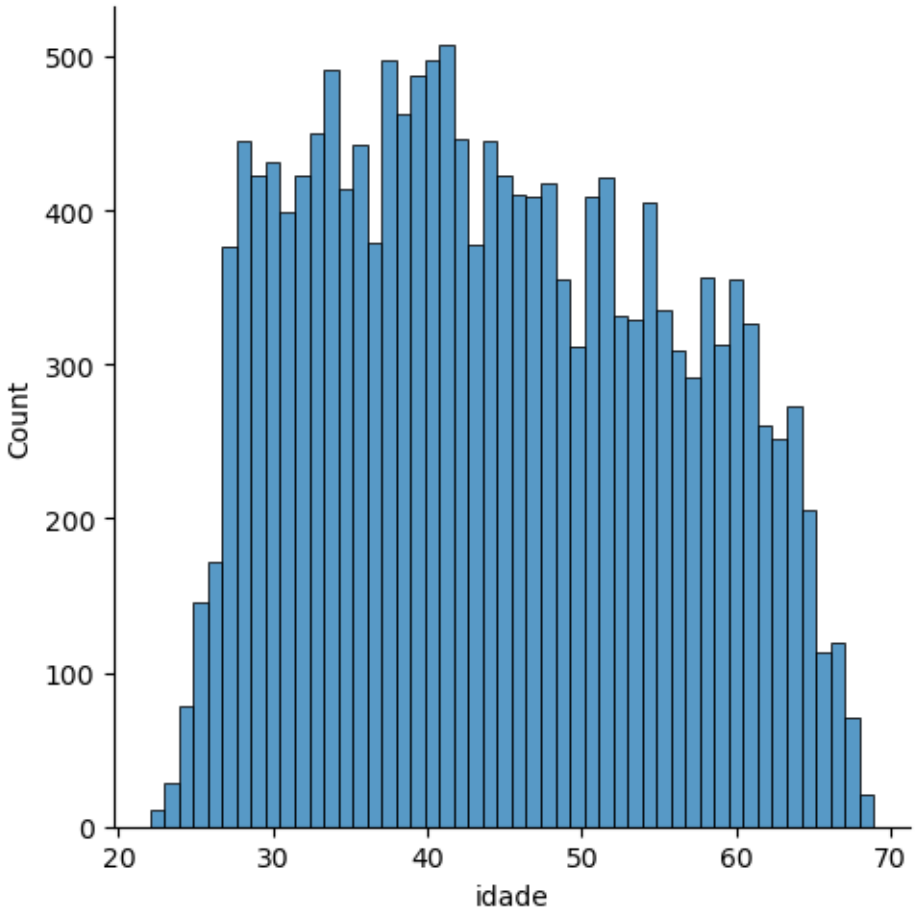
Com base no código da célula anterior, construa o histograma da variavel `dade`.

[10]:
```
plt.clf()
var = "idade"

sns.displot(df, x = var, bins = 50)
plt.show()
```

```
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages/seaborn/axisgrid.py:118: UserWarning: The figure layout has changed to
tight
  self._figure.tight_layout(*args, **kwargs)
```

```
<Figure size 640x480 with 0 Axes>
```

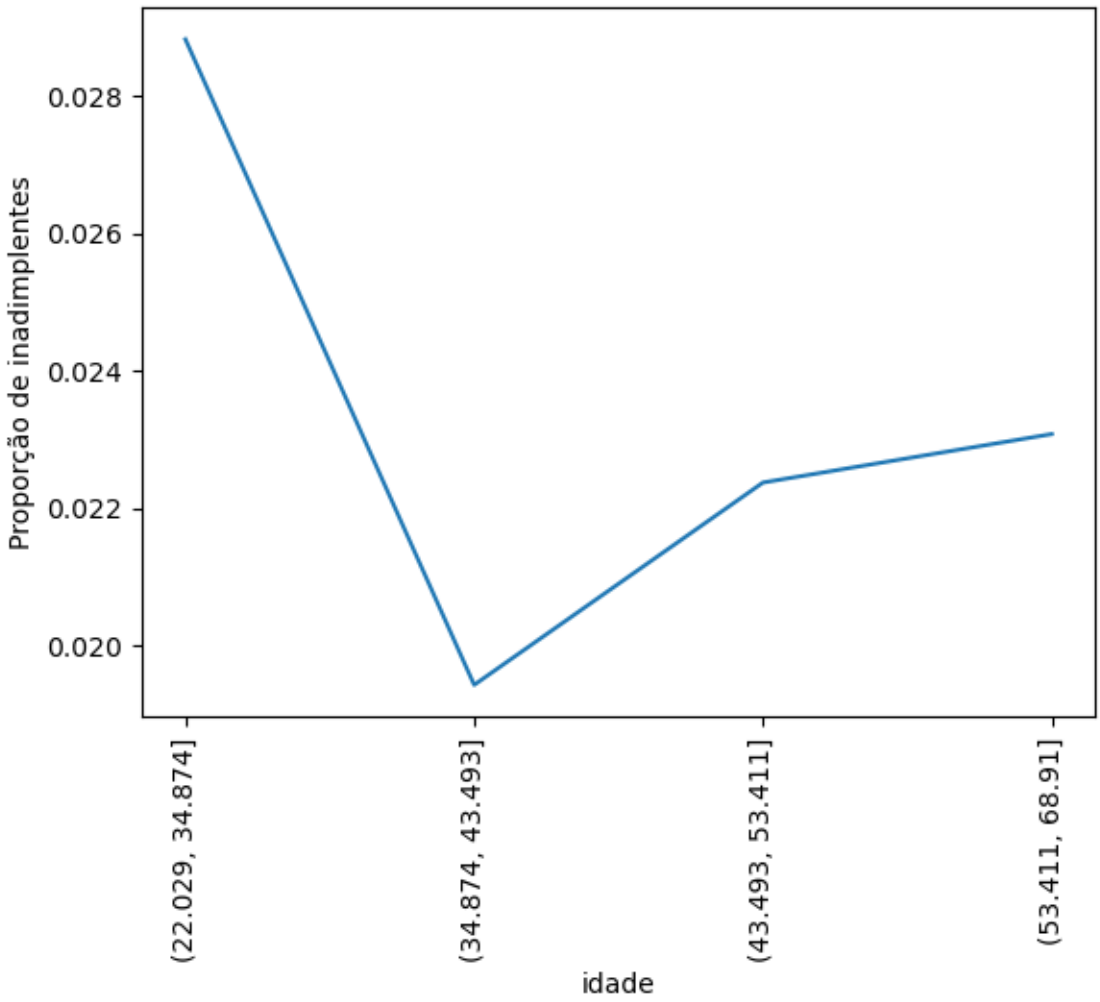


1.2.3 Entendimento dos dados - Bivariadas

Entender a alteração da inadimplência indicada pela variável resposta (`AtrasoRelevante2anos`) e as variáveis explicativas (demais). Para isto, vamos calcular a taxa de inadimplentes (qtd inadimplentes / total) para diferentes grupos definidos pelas variáveis explicativas.

[11]:
```
var = 'idade'
cat_srs, bins = pd.qcut(df[var], 4, retbins=True)
g = df.groupby(cat_srs)
biv = g['mau'].mean()

ax = biv.plot.line()
ax.set_ylabel("Proporção de inadimplentes")
ticks = plt.xticks(range(len(biv.index.values)), biv.index.values, rotation =␣
↪90)
```

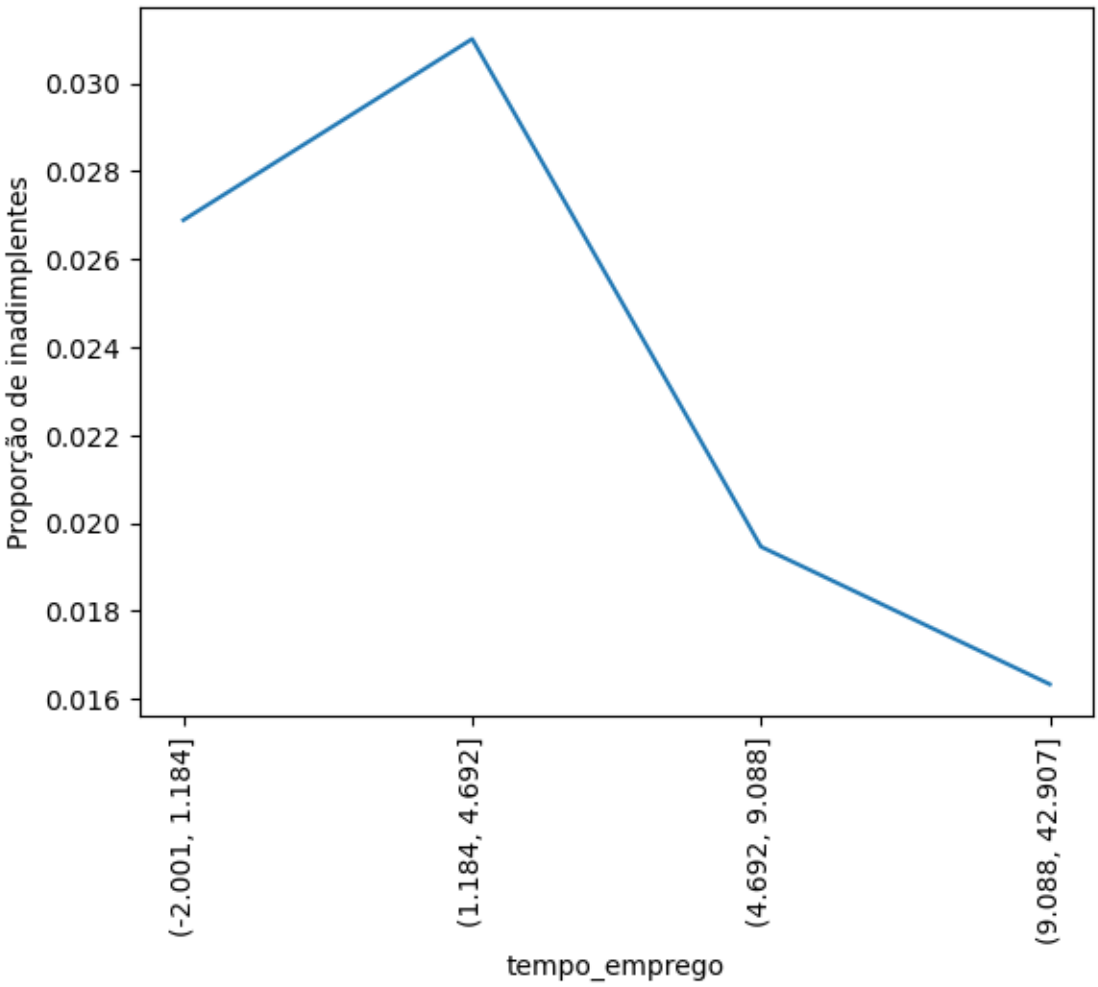


1.2.4 Tarefa 03 - Bivariada

Com base no código da célula anterior, construa uma análise bivariada para a variável `tempo_emprego`. Em seguida, insira uma célula de markdown e conclua se a variável parece discriminar risco de crédito.

[12]:
```
var = 'tempo_emprego'
cat_srs, bins = pd.qcut(df[var], 4, retbins=True)
g = df.groupby(cat_srs)
biv = g['mau'].mean()

ax = biv.plot.line()
ax.set_ylabel("Proporção de inadimplentes")
ticks = plt.xticks(range(len(biv.index.values)), biv.index.values, rotation =␣
↪90)
```



Conclusão Podemos concluir que clientes com tempo de emprego menor do que 4 anos têm maior proporção de inadimplentes. A inadimplência diminui consistentemente a partir do 4 ano.

1.3 Etapa 3 Crisp-DM: Preparação dos dados

Nessa etapa realizamos tipicamente as seguintes operações com os dados: - seleção Neste caso, os dados já estão pré-selecionados - limpeza Precisaremos identificar e tratar dados faltantes - construção Neste primeiro exercício não faremos construção de novas variáveis - integração Temos apenas uma fonte de dados, não é necessário agregação - formatação Os dados já se encontram em formatos úteis

Os dados já estão pré-selecionados, construídos e integrados, mas há dados faltantes que serão eliminados na próxima célula

[13]:
```
metadata = pd.DataFrame(df.dtypes, columns = ['tipo'])

metadata['n_categorias'] = 0
```

```
for var in metadata.index:
    metadata.loc[var,'n_categorias'] = len(df.groupby([var]).size())

metadata
```

[13]:

	tipo	n_categorias
sexo	object	2
posse_de_veiculo	object	2
posse_de_imovel	object	2
qtd_filhos	int64	8
tipo_renda	object	5
educacao	object	5
estado_civil	object	5
tipo_residencia	object	6
idade	float64	5298
tempo_emprego	float64	3005
possui_celular	int64	1
possui_fone_comercial	int64	2
possui_fone	int64	2
possui_email	int64	2
qt_pessoas_residencia	float64	9
mau	bool	2

[14]:
```
def convert_dummy(df, feature,rank=0):
    pos = pd.get_dummies(df[feature], prefix=feature)
    mode = df[feature].value_counts().index[rank]
    biggest = feature + '_' + str(mode)
    pos.drop([biggest],axis=1,inplace=True)
    df.drop([feature],axis=1,inplace=True)
    df=df.join(pos)
    return df
```

[15]:
```
for var in metadata[metadata['tipo'] == 'object'].index:
    df = convert_dummy(df, var)
```

[16]:
```
df
```

[16]:

	qtd_filhos	idade	tempo_emprego	possui_celular	\
0	0	58.832877	3.106849	1	
1	0	52.356164	8.358904	1	
2	0	52.356164	8.358904	1	
3	0	46.224658	2.106849	1	
4	0	29.230137	3.021918	1	
…	…	…	…	…	
16645	0	54.109589	9.884932	1	
16646	0	43.389041	7.380822	1	

```
16647           0  30.005479       9.800000               1
16648           0  30.005479       9.800000               1
16649           0  33.936986       3.630137               1

       possui_fone_comercial  possui_fone  possui_email  \
0                          0            0             0
1                          0            1             1
2                          0            1             1
3                          1            1             1
4                          0            0             0
…                        …            …             …
16645                      0            0             0
16646                      1            1             0
16647                      1            0             0
16648                      1            0             0
16649                      0            1             1

       qt_pessoas_residencia    mau  sexo_M  …  educacao_Lower secondary  \
0                        2.0  False    True  …                    False
1                        1.0  False   False  …                    False
2                        1.0  False   False  …                    False
3                        2.0  False    True  …                    False
4                        2.0  False   False  …                    False
…                        …    …     …  …                      …
16645                    2.0   True   False  …                    False
16646                    2.0   True   False  …                    False
16647                    2.0   True    True  …                    False
16648                    2.0   True    True  …                    False
16649                    2.0   True   False  …                    False

       estado_civil_Civil marriage  estado_civil_Separated  \
0                            False                   False
1                            False                   False
2                            False                   False
3                            False                   False
4                            False                   False
…                            …                     …
16645                         True                   False
16646                        False                   False
16647                        False                   False
16648                        False                   False
16649                        False                   False

       estado_civil_Single / not married  estado_civil_Widow  \
0                                  False               False
1                                   True               False
2                                   True               False
```

```
3                                False              False
4                                False              False
…                                …                  …
16645                            False              False
16646                            False              False
16647                            False              False
16648                            False              False
16649                            False              False

       tipo_residencia_Co-op apartment  tipo_residencia_Municipal apartment  \
0                                False                                False
1                                False                                False
2                                False                                False
3                                False                                False
4                                False                                False
…                                …                                  …
16645                            False                                False
16646                            False                                False
16647                            False                                False
16648                            False                                False
16649                            False                                False

       tipo_residencia_Office apartment  tipo_residencia_Rented apartment  \
0                                False                               False
1                                False                               False
2                                False                               False
3                                False                               False
4                                False                               False
…                                …                                 …
16645                            False                               False
16646                            False                               False
16647                            False                               False
16648                            False                               False
16649                            False                               False

       tipo_residencia_With parents
0                             False
1                             False
2                             False
3                             False
4                             False
…                             …
16645                         False
16646                         False
16647                         False
16648                         False
16649                         False
```

```
[16650 rows x 29 columns]
```

1.4 Etapa 4 Crisp-DM: Modelagem

Nessa etapa que realizaremos a construção do modelo. Os passos típicos são: - Selecionar a técnica de modelagem Utilizaremos a técnica de floresta aleatória (**random forest**), pois é uma técnica bastante versátil e robusta que captura bem padrões complexos nos dados, relativamente fácil de se usar e que costuma produzir excelentes resultados para uma classificação como estas. Vamos ver esse algoritmo em detalhes mais adiante no curso, mas pense nele por enquanto como uma regra complexa baseada nas variáveis explicativas que classifica o indivíduo como inadimplente ou não. Mais adiante no curso vamos extrair mais dessa técnica. - Desenho do teste Antes de rodar o modelo precisamos construir um desenho do teste que será realizado. Para desenvolver um modelo como este, é considerado uma boa prática dividir a base em duas, uma chamada `treinamento`, onde o algoritmo 'aprende', e outra chamada `teste`, onde o algoritmo é avaliado. Essa prática fornece uma métrica de avaliação mais fidedigna do algoritmo, falaremos mais detalhes em lições futuras. - Avaliação do modelo Faremos a avaliação do nosso modelo através do percentual de acerto, avaliando a classificação do modelo (inadimplente e não inadimplente) e comparando com o estado real armazenado na variável resposta (`AtrasoRelevante2anos`). Esse percentual de acerto é frequentemente chamado de acurácia (**obs:** nunca usar assertividade... a**ss**ertivo não é aquele que a**c**erta, e sim "*adj.: em que o locutor declara algo, positivo ou negativo, do qual assume inteiramente a validade; declarativo.*" a**C**ertivo está errado ;) #### Dividindo a base em treino e teste

[17]:
```
# Tirando a v. resposta da base de treinamento
x = df.drop("mau",axis = 1)
y = df["mau"]

# Tirando ID da base de treinamento e teste
x_train, x_test, y_train, y_test = train_test_split(x, y)
```

[18]:
```
x_train
```

[18]:

	qtd_filhos	idade	tempo_emprego	possui_celular	\
7064	0	33.147945	4.293151	1	
7091	0	40.561644	4.964384	1	
10046	3	40.652055	4.216438	1	
12822	0	62.161644	-2.000000	1	
2531	0	60.183562	-2.000000	1	
…	…	…	…	…	
9661	0	65.123288	-2.000000	1	
3405	1	30.558904	12.079452	1	
10547	0	35.742466	1.838356	1	
7504	0	39.457534	2.901370	1	
7856	0	45.317808	2.972603	1	

	possui_fone_comercial	possui_fone	possui_email	\
7064	0	0	0	

```
7091                        0            1             0
10046                       0            0             0
12822                       0            0             0
2531                        0            0             0
…                         …          …           …
9661                        0            0             0
3405                        0            0             0
10547                       0            0             0
7504                        0            0             1
7856                        0            1             0

       qt_pessoas_residencia  sexo_M  posse_de_veiculo_Y  …  \
7064                     1.0    True                True  …
7091                     1.0   False               False  …
10046                    5.0   False               False  …
12822                    2.0   False               False  …
2531                     2.0   False               False  …
…                        …     …                   …  …
9661                     2.0   False               False  …
3405                     3.0    True                True  …
10547                    2.0   False               False  …
7504                     2.0   False               False  …
7856                     2.0   False                True  …

       educacao_Lower secondary  estado_civil_Civil marriage  \
7064                      False                        False
7091                      False                        False
10046                     False                        False
12822                     False                        False
2531                      False                         True
…                         …                          …
9661                      False                        False
3405                      False                        False
10547                     False                         True
7504                      False                        False
7856                      False                        False

       estado_civil_Separated  estado_civil_Single / not married  \
7064                    False                               True
7091                     True                              False
10046                   False                              False
12822                   False                              False
2531                    False                              False
…                       …                                …
9661                    False                              False
3405                    False                              False
10547                   False                              False
```

```
7504                    False                                False
7856                    False                                False

       estado_civil_Widow  tipo_residencia_Co-op apartment  \
7064                False                            False
7091                False                            False
10046               False                            False
12822               False                            False
2531                False                            False
…                   …                                …
9661                False                            False
3405                False                            False
10547               False                            False
7504                False                            False
7856                False                            False

       tipo_residencia_Municipal apartment  tipo_residencia_Office apartment  \
7064                                  True                               False
7091                                 False                               False
10046                                False                               False
12822                                False                               False
2531                                 False                               False
…                                    …                                   …
9661                                 False                               False
3405                                 False                                True
10547                                False                               False
7504                                 False                               False
7856                                 False                               False

       tipo_residencia_Rented apartment  tipo_residencia_With parents
7064                               False                        False
7091                               False                         True
10046                              False                        False
12822                              False                        False
2531                               False                        False
…                                  …                            …
9661                               False                        False
3405                               False                        False
10547                              False                        False
7504                               False                        False
7856                               False                        False

[12487 rows x 28 columns]
```

1.4.1 Rodando o modelo

A função RandomForestClassifier gera a estrutura da floresta aleatória, e o parâmetro `n_estimator` define o número de árvores na floresta. Normalmente a acurácia do modelo tende a aumentar com o número de árvores, pelo menos até um certo limite - e aumenta também o recurso computacional demandado. Você pode alterar esse parâmetro e verificar se a acurácia do seu modelo melhora - não recomendamos valores muito altos. Vá alterando aos poucos e percebendo como o tempo aumenta com os seus recursos. Não é necessário ir muito além de umas 100 árvores.

[22]:
```
# Treinar uma Random Forest com 5 árvores

clf = RandomForestClassifier(n_estimators=2)
clf.fit(x_train,y_train)
```

[22]:
```
RandomForestClassifier(n_estimators=2)
```

[23]:
```
# Calculando a acuracia

y_pred = clf.predict(x_test)
acc = metrics.accuracy_score(y_test, y_pred)
print('Acurácia: {0:.2f}%'.format(acc*100))
```

```
Acurácia: 97.33%
```

[25]:
```
print(tab)
print(tab.index)
print(tab.columns)
```

```
mau    False  True
row_0
False   4036     88
True      23     16
Index([False, True], dtype='bool', name='row_0')
Index([False, True], dtype='bool', name='mau')
```

[28]:
```
# Matriz de confusão

tab = pd.crosstab(index = y_pred, columns = y_test)
print((tab[False][False] / (tab[False][False] + tab[True][False])))  #␣
 ↪Especificidade
print((tab[True][True] / (tab[False][True] + tab[True][True])))      #␣
 ↪Sensibilidade (Recall)
tab
```

```
0.9786614936954413
0.41025641025641024
```

[28]:
```
mau    False  True
row_0
False   4036     88
True      23     16
```

1.4.2 Tarefa 04 - Bivariada

Para essa tarefa, crie três células extras, copie nelas o código das três células anteriores (na mesma ordem) e altere o parâmetro `n_estimators=` da função `RandomForestClassifier` e insira uma quantidade maior que 3 nesse parâmetro. Rode as três células anteriores para calcular a acurácia do modelo e veja se você consegue uma acurácia melhor.

[31]:
```
# Treinar uma Random Forest com 5 árvores

clf = RandomForestClassifier(n_estimators=50)

clf.fit(x_train,y_train)
```

[31]:
```
RandomForestClassifier(n_estimators=50)
```

[32]:
```
# Calculando a acuracia

y_pred = clf.predict(x_test)
acc = metrics.accuracy_score(y_test, y_pred)
print('Acurácia: {0:.2f}%'.format(acc*100))
```

```
Acurácia: 97.65%
```

[33]:
```
# Matriz de confusão

tab = pd.crosstab(index = y_pred, columns = y_test)
print((tab[False][False] / (tab[False][False] + tab[True][False])))  #␣
 ↪Especificidade
print((tab[True][True] / (tab[False][True] + tab[True][True])))      #␣
 ↪Sensibilidade (Recall)
tab
```

```
0.9810541656546029
0.5652173913043478
```

[33]:
```
mau    False  True
row_0
False   4039     78
True      20     26
```

1.5 Etapa 5 Crisp-DM: Avaliação dos resultados

A etapa final do CRISP. Neste casp, a nossa avaliação termina com a acurácia. Mas em problemas futuros aprofundaremos mais - a ideia seria avaliar o impacto do uso do modelo no negócio, ou seja, o quanto o resultado financeiro melhora em detrimento da utilização do modelo.

Como um exemplo simples, considere que um cliente bom pagador deixa (em média) 5 '*dinheiros*' de lucro, e um mau pagador deixa (em média) 100 '*dinheiros*' de prejuízo.

de acordo com a matriz de confusão:

Decisão	lucro dos bons	lucro dos maus	total
Aprovador	4042 x 5	72 x (-100)	13.010
Reprovar	27 x 5	22 x (-100)	-2.065

Estariamos evitando, portanto, um prejuízo de -2.145 '*dinheiros*' - o que na prática significa um aumento no lucro.

1.6 Etapa 6 Crisp-DM: Implantação

Nessa etapa colocamos em uso o modelo desenvolvido, normalmente implementando o modelo desenvolvido em um motor de crédito que toma as decisões com algum nível de automação - tipicamente aprovando automaticamente clientes muito bons, negando automaticamente clientes muito ruins, e enviando os intermediários para análise manual.