Módulo | Python: Projeto Final

Caderno de **Aula**
Professor André Perez

Tópicos

1. Introdução ao Kaggle;
2. Exploração de dados;
3. Transformação e limpeza de dados;
4. Visualização de dados;
5. Storytelling.

Aulas

1. Introdução ao Kaggle

Kaggle é a maior comunidade online de ciência de dados e aprendizado de máquina. A plataforma permite que usuários encontrem e publiquem conjuntos de **dados**, construam e compartilhem **notebooks** (como este do Google Colab) e participem de **competições** (que pagam muito dinheiro as vezes) e desafios de dados.

> Vamos publicar nosso notebook de exercícios na plataforma web do Kaggle para que você possa compartilhar tudo o que você aprendeu neste curso e compor o seu portfólio.

2. Exploração de Dados

Vamos explorar dados de crédito presentes neste neste link. Os dados estão no formato CSV e contém informações sobre clientes de uma instituição financeira. Em especial, estamos interessados em explicar a segunda coluna, chamada de **default**, que indica se um cliente é adimplente(`default = 0`), ou inadimplente (`default = 1`), ou seja, queremos entender o porque um cliente deixa de honrar com suas dívidas baseado no comportamento de outros

atributos, como salário, escolaridade e movimentação financeira. Uma descrição completa dos atributos está abaixo.

> O atributo de interesse (`default`) é conhecido como **variável resposta** ou **variável dependente**, já os demais atributos que buscam explicá-la (`idade` , `salário` , etc.) são conhecidas como **variáveis explicatívas**, **variáveis independentes** ou até **variáveis preditoras**.

Coluna	Descrição
id	Número da conta
default	Indica se o cliente é adimplente (0) ou inadimplente (1)
idade	---
sexo	---
depedentes	---
escolaridade	---
estado_civil	---
salario_anual	Faixa do salario mensal multiplicado por 12
tipo_cartao	Categoria do cartao: blue, silver, gold e platinium
meses_de_relacionamento	Quantidade de meses desde a abertura da conta
qtd_produtos	Quantidade de produtos contratados
iteracoes_12m	Quantidade de iteracoes com o cliente no último ano
meses_inatico_12m	Quantidade de meses que o cliente ficou inativo no último ano
limite_credito	Valor do limite do cartão de crédito
valor_transacoes_12m	Soma total do valor das transações no cartão de crédito no último ano
qtd_transacoes_12m	Quantidade total de transações no cartão de crédito no último ano

Vamos começar lendos os dados num dataframe `pandas` .

In [ ]:
```
import pandas as pd
```

In [ ]:
```
df = pd.read_csv(
    'https://raw.githubusercontent.com/andre-marcos-perez/' + \
    'ebac-course-utils/develop/dataset/credito.csv',
    na_values='na'
)
```

In [ ]:
```
df.head(n=10)
```

Com o dados em mãos, vamos conhecer um pouco melhor a estrutura do nosso conjunto de dados.

2.1. Estrutura

In [ ]:
```
df.shape # retorna uma tupla (qtd linhas, qtd colunas)
```

In [ ]:
```

```

```
df[df['default'] == 0].shape
```

In [ ]:
```
df[df['default'] == 1].shape
```

In [ ]:
```
qtd_total, _ = df.shape
qtd_adimplentes, _ = df[df['default'] == 0].shape
qtd_inadimplentes, _ = df[df['default'] == 1].shape
```

In [ ]:
```
print(f"A proporcão clientes adimplentes é de " + \
      f"{round(100 * qtd_adimplentes / qtd_total, 2)}%"
     )

print(f"A proporcão clientes inadimplentes é de " + \
      f"{round(100 * qtd_inadimplentes / qtd_total, 2)}%"
     )
```

2.2. Schema

In [ ]:
```
df.head(n=5)
```

- Colunas e seus respectivos tipos de dados.

In [ ]:
```
df.dtypes
```

- Atributos **categóricos**.

In [ ]:
```
df.select_dtypes('object').describe().transpose()
```

- Atributos **numéricos**.

In [ ]:
```
df.drop('id', axis=1).select_dtypes('number').describe().transpose()
```

2.3. Dados faltantes

Dados faltantes podem ser:

- Vazios (`""`);
- Nulos (`None`);
- Não disponíveis ou aplicaveis (`na` , `NA` , etc.);
- Não numérico (`nan` , `NaN` , `NAN` , etc).

In [ ]:
```
df.head()
```

Podemos verificar quais colunas possuem dados faltantes.

In [ ]:
```
df.isna().any()
```

- A função abaixo levanta algumas estatisticas sobre as colunas dos dados faltantes.

In [ ]:
```
def stats_dados_faltantes(df: pd.DataFrame) -> None:

  stats_dados_faltantes = []
  for col in df.columns:
    if df[col].isna().any():
      qtd, _ = df[df[col].isna()].shape
      total, _ = df.shape
      dict_dados_faltantes = {col:
        {
          'quantidade': qtd,
          "porcentagem": round(100 * qtd/total, 2)
        }
      }
      stats_dados_faltantes.append(dict_dados_faltantes)

  for stat in stats_dados_faltantes:
    print(stat)
```

In [ ]:
```
stats_dados_faltantes(df=df)
```

In [ ]:
```
stats_dados_faltantes(df=df[df['default'] == 0])
```

In [ ]:
```
stats_dados_faltantes(df=df[df['default'] == 1])
```

3. Transformação e limpeza de dados

Agora que conhecemos melhor a natureza do nosso conjunto de dados, vamos conduzir uma atividade conhecida como *data wrangling* que consiste na transformação e limpeza dos dados do conjunto para que possam ser melhor analisados. Em especial, vamos remover:

- Corrigir o *schema* das nossas colunas;
- Remover os dados faltantes.

3.1. Correção de schema

Na etapa de exploração, notamos que as colunas **limite_credito** e **valor_transacoes_12m** estavam sendo interpretadas como colunas categóricas (`dtype = object`).

In [ ]:
```
df[['limite_credito', 'valor_transacoes_12m']].dtypes
```

In [ ]:
```
df[['limite_credito', 'valor_transacoes_12m']].head(n=5)
```

Vamos criar uma função `lambda` para limpar os dados. Mas antes, vamos testar sua aplicação através do método funcional `map` :

In [ ]:
```
fn = lambda valor: float(valor.replace(".", "").replace(",", "."))

valores_originais = [
    '12.691,51',
    '8.256,96',
    '3.418,56',
```

```
    '3.313,03',
    '4.716,22'
]

valores_limpos = list(map(fn, valores_originais))

print(valores_originais)
print(valores_limpos)
```

Com a função `lambda` de limpeza pronta, basta aplica-la nas colunas de interesse.

In [ ]:
```
df['valor_transacoes_12m'] = df['valor_transacoes_12m'].apply(fn)
df['limite_credito'] = df['limite_credito'].apply(fn)
```

Vamos descrever novamente o *schema*:

In [ ]:
```
df.dtypes
```

- Atributos **categóricos**.

In [ ]:
```
df.select_dtypes('object').describe().transpose()
```

- Atributos **numéricos**.

In [ ]:
```
df.drop('id', axis=1).select_dtypes('number').describe().transpose()
```

3.2. Remoção de dados faltantes

Como o pandas está ciente do que é um dados faltante, a remoção das linhas problemáticas é trivial.

In [ ]:
```
df.dropna(inplace=True)
```

Vamos analisar a estrutura dos dados novamente.

In [ ]:
```
df.shape
```

In [ ]:
```
df[df['default'] == 0].shape
```

In [ ]:
```
df[df['default'] == 1].shape
```

In [ ]:
```
qtd_total_novo, _ = df.shape
qtd_adimplentes_novo, _ = df[df['default'] == 0].shape
qtd_inadimplentes_novo, _ = df[df['default'] == 1].shape
```

In [ ]:
```
print(f"A proporcão adimplentes ativos é de " + \
      f"{round(100 * qtd_adimplentes / qtd_total, 2)}%"]
     )

print(f"A nova proporcão de clientes adimplentes é de " + \
```

```
      f"{round(100 * qtd_adimplentes_novo / qtd_total_novo, 2)}%"
     )

print("")

print(f"A proporcão clientes inadimplentes é de " + \
      f"{round(100 * qtd_inadimplentes / qtd_total, 2)}%"
     )

print(f"A nova proporcão de clientes inadimplentes é de " + \
      f"{round(100 * qtd_inadimplentes_novo / qtd_total_novo, 2)}%"
     )
```

4. Visualização de dados

Os dados estão prontos, vamos criar diversas visualizações para correlacionar variáveis explicativas com a variável resposta para buscar entender qual fator leva um cliente a inadimplencia. E para isso, vamos sempre comparar a base com todos os clientes com a base de adimplentes e inadimplentes.

Começamos então importando os pacotes de visualização e separando os clientes adimplentes e inadimplentes

In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt

sns.set_style("whitegrid")
```

In [ ]:
```
df_adimplente = df[df['default'] == 0]
```

In [ ]:
```
df_inadimplente = df[df['default'] == 1]
```

4.1. Visualizações categóricas

Nesta seção, vamos visualizar a relação entre a variável resposta **default** com os atributos categóricos.

In [ ]:
```
df.select_dtypes('object').head(n=5)
```

- Escolaridade

In [ ]:
```
coluna = 'escolaridade'
titulos = [
    'Escolaridade dos Clientes',
    'Escolaridade dos Clientes Adimplentes',
    'Escolaridade dos Clientes Inadimplentes'
]

eixo = 0
max_y = 0
max = df.select_dtypes('object').describe()[coluna]['freq'] * 1.1

figura, eixos = plt.subplots(1,3, figsize=(20, 5), sharex=True)
```

```
for dataframe in [df, df_adimplente, df_inadimplente]:

  df_to_plot = dataframe[coluna].value_counts().to_frame()
  df_to_plot.rename(columns={coluna: 'frequencia_absoluta'}, inplace=True)
  df_to_plot[coluna] = df_to_plot.index
  df_to_plot.sort_values(by=[coluna], inplace=True)
  df_to_plot.sort_values(by=[coluna])

  f = sns.barplot(
      x=df_to_plot[coluna],
      y=df_to_plot['frequencia_absoluta'],
      ax=eixos[eixo]
  )
  f.set(
      title=titulos[eixo],
      xlabel=coluna.capitalize(),
      ylabel='Frequência Absoluta'
  )
  f.set_xticklabels(labels=f.get_xticklabels(), rotation=90)

  _, max_y_f = f.get_ylim()
  max_y = max_y_f if max_y_f > max_y else max_y
  f.set(ylim=(0, max_y))

  eixo += 1

figura.show()
```

- Salário Anual

In [ ]:

```
coluna = 'salario_anual'
titulos = [
    'Salário Anual dos Clientes',
    'Salário Anual dos Clientes Adimplentes',
    'Salário Anual dos Clientes Inadimplentes'
]

eixo = 0
max_y = 0
figura, eixos = plt.subplots(1,3, figsize=(20, 5), sharex=True)

for dataframe in [df, df_adimplente, df_inadimplente]:

  df_to_plot = dataframe[coluna].value_counts().to_frame()
  df_to_plot.rename(columns={coluna: 'frequencia_absoluta'}, inplace=True)
  df_to_plot[coluna] = df_to_plot.index
  df_to_plot.reset_index(inplace=True, drop=True)
  df_to_plot.sort_values(by=[coluna], inplace=True)

  f = sns.barplot(
      x=df_to_plot[coluna],
      y=df_to_plot['frequencia_absoluta'],
      ax=eixos[eixo]
  )
  f.set(
      title=titulos[eixo],
      xlabel=coluna.capitalize(),
      ylabel='Frequência Absoluta'
  )
  f.set_xticklabels(labels=f.get_xticklabels(), rotation=90)
  _, max_y_f = f.get_ylim()
  max_y = max_y_f if max_y_f > max_y else max_y
  f.set(ylim=(0, max_y))
```

```
  eixo += 1

figura.show()
```

4.2. Visualizações numéricas

Nesta seção, vamos visualizar a relação entre a variável resposta **default** com os atributos numéricos.

In [ ]:
```
df.drop(['id', 'default'], axis=1).select_dtypes('number').head(n=5)
```

- Quantidade de Transações nos Últimos 12 Meses

In [ ]:
```
coluna = 'qtd_transacoes_12m'
titulos = [
    'Qtd. de Transações no Último Ano',
    'Qtd. de Transações no Último Ano de Adimplentes',
    'Qtd. de Transações no Último Ano de Inadimplentes'
]

eixo = 0
max_y = 0
figura, eixos = plt.subplots(1,3, figsize=(20, 5), sharex=True)

for dataframe in [df, df_adimplente, df_inadimplente]:

  f = sns.histplot(x=coluna, data=dataframe, stat='count', ax=eixos[eixo])
  f.set(
      title=titulos[eixo],
      xlabel=coluna.capitalize(),
      ylabel='Frequência Absoluta'
  )

  _, max_y_f = f.get_ylim()
  max_y = max_y_f if max_y_f > max_y else max_y
  f.set(ylim=(0, max_y))

  eixo += 1

figura.show()
```

- Valor das Transações nos Últimos 12 Meses

In [ ]:
```
coluna = 'valor_transacoes_12m'
titulos = [
    'Valor das Transações no Último Ano',
    'Valor das Transações no Último Ano de Adimplentes',
    'Valor das Transações no Último Ano de Inadimplentes'
]

eixo = 0
max_y = 0
figura, eixos = plt.subplots(1,3, figsize=(20, 5), sharex=True)

for dataframe in [df, df_adimplente, df_inadimplente]:

  f = sns.histplot(x=coluna, data=dataframe, stat='count', ax=eixos[eixo])
  f.set(
      title=titulos[eixo],
```

```
        xlabel=coluna.capitalize(),
        ylabel='Frequência Absoluta'
    )

    _, max_y_f = f.get_ylim()
    max_y = max_y_f if max_y_f > max_y else max_y
    f.set(ylim=(0, max_y))

    eixo += 1

figura.show()
```

- Valor de Transações nos Últimos 12 Meses x Quantidade de Transações nos Últimos 12 Meses

In [ ]:
```
f = sns.relplot(
    x='valor_transacoes_12m',
    y='qtd_transacoes_12m',
    data=df,
    hue='default'
)

_ = f.set(
    title='Relação entre Valor e Quantidade de Transações no Último Ano',
    xlabel='Valor das Transações no Último Ano',
    ylabel='Quantidade das Transações no Último Ano'
  )
```

5. Storytelling

O *storytelling* no contexto de dados é um técnica de apresentação de resultados orientado a dados, ou seja, contar uma história baseada nos *insights* que foram gerados através da análise dos dados. Notebooks como este do Google Colab e os do Kaggle são excelentes ferramentas para conduzir e compartilhar *storytelling* de dados devido a natureza texto-código de suas céluas.

Para você montar o seu portifólio, eu sugiro a construção de um notebook com a seguinte estrutura (vou disponibiliza-la nos exercícios):

1. Título;
2. Breve descrição do problema;
3. Código de importação de bibliotecas;
4. Código o download/carregamento/geração de dados;
5. Etapa de exploração;
6. Etapa de limpeza e transformação;
7. Etapa de análise (com visualizações);
8. Resumo dos *insights* gerados.

Busquei organizar este notebook desta forma. Ademais, os notebooks presentes na plataforma do Kaggle são excelentes exemplos a serem seguidos, em especial os primeiros colocados em competições.

Para finalizar, algumas dicas:

1. Estruture seu código sempre de acordo com as boas práticas PEP8, assim ele será mais legível para o leitor;
2. Sempre se preocupe com a aparência dos seus gráficos, todos devem ter (no mínimo) título no topo e nos eixos;
3. Use e abuso das células de texto para estruturar seu notebook, siga as mesmas técnicas que eu utilizo nos notebooks do curso para estruturar seu texto.

Quando terminar, se possível, entra em contato comigo, eu adoraria ver o seu notebook final pronto! Meu LinkedIn está no topo da página, logo após o logo da EBAC. Espero que você tenha gostado, foi um prazer trabalhar com você!

Até a próxima.