Untitled

December 9, 2023

1 Projeto Python IA: Inteligência Artificial e Previsões

1.0.1 Case: Score de Crédito dos Clientes

Você foi contratado por um banco para conseguir definir o score de crédito dos clientes. Você precisa analisar todos os clientes do banco e, com base nessa análise, criar um modelo que consiga ler as informações do cliente e dizer automaticamente o score de crédito dele: Ruim, Ok, Bom

[1]:
```
import pandas as pd

tabela = pd.read_csv("clientes.csv") # importa a base de dados
display(tabela)
```

```
       id_cliente  mes  idade  profissao  salario_anual  num_contas  \
0            3392    1   23.0  cientista       19114.12         3.0
1            3392    2   23.0  cientista       19114.12         3.0
2            3392    3   23.0  cientista       19114.12         3.0
3            3392    4   23.0  cientista       19114.12         3.0
4            3392    5   23.0  cientista       19114.12         3.0
…             …  …    …         …              …            …
99995       37932    4   25.0   mecanico       39628.99         4.0
99996       37932    5   25.0   mecanico       39628.99         4.0
99997       37932    6   25.0   mecanico       39628.99         4.0
99998       37932    7   25.0   mecanico       39628.99         4.0
99999       37932    8   25.0   mecanico       39628.99         4.0

       num_cartoes  juros_emprestimo  num_emprestimos  dias_atraso  …  \
0              4.0               3.0              4.0          3.0  …
1              4.0               3.0              4.0          3.0  …
2              4.0               3.0              4.0          3.0  …
3              4.0               3.0              4.0          5.0  …
4              4.0               3.0              4.0          6.0  …
…             …                 …                …            …  …
99995          6.0               7.0              2.0         23.0  …
99996          6.0               7.0              2.0         18.0  …
99997          6.0               7.0              2.0         27.0  …
99998          6.0               7.0              2.0         20.0  …
99999          6.0               7.0              2.0         18.0  …
```

```
       idade_historico_credito  investimento_mensal  \
0                        265.0            21.465380
1                        266.0            21.465380
2                        267.0            21.465380
3                        268.0            21.465380
4                        269.0            21.465380
…                          …                …
99995                    378.0            24.028477
99996                    379.0            24.028477
99997                    380.0            24.028477
99998                    381.0            24.028477
99999                    382.0            24.028477

           comportamento_pagamento  saldo_final_mes  score_credito  \
0      alto_gasto_pagamento_baixos       312.494089           Good
1       baixo_gasto_pagamento_alto       284.629162           Good
2      baixo_gasto_pagamento_medio       331.209863           Good
3      baixo_gasto_pagamento_baixo       223.451310           Good
4       alto_gasto_pagamento_medio       341.489231           Good
…                              …             …            …
99995    alto_gasto_pagamento_alto       479.866228           Poor
99996   alto_gasto_pagamento_medio       496.651610           Poor
99997    alto_gasto_pagamento_alto       516.809083           Poor
99998   baixo_gasto_pagamento_alto       319.164979       Standard
99999   alto_gasto_pagamento_medio       393.673696           Poor

       emprestimo_carro  emprestimo_casa emprestimo_pessoal  \
0                     1                1                  1
1                     1                1                  1
2                     1                1                  1
3                     1                1                  1
4                     1                1                  1
…                   …                …                  …
99995                 1                0                  0
99996                 1                0                  0
99997                 1                0                  0
99998                 1                0                  0
99999                 1                0                  0

       emprestimo_credito emprestimo_estudantil
0                       1                     0
1                       1                     0
2                       1                     0
3                       1                     0
4                       1                     0
…                     …                     …
99995                   0                     1
99996                   0                     1
```

```
99997                   0                     1
99998                   0                     1
99999                   0                     1

[100000 rows x 25 columns]
```

[2]:
```
# verificar se temos valores vazios ou valores reconhecidos em formato errado
print(tabela.info())
print(tabela.columns)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 25 columns):
 #   Column                     Non-Null Count   Dtype
---  ------                     --------------   -----
 0   id_cliente                 100000 non-null  int64
 1   mes                        100000 non-null  int64
 2   idade                      100000 non-null  float64
 3   profissao                  100000 non-null  object
 4   salario_anual              100000 non-null  float64
 5   num_contas                 100000 non-null  float64
 6   num_cartoes                100000 non-null  float64
 7   juros_emprestimo           100000 non-null  float64
 8   num_emprestimos            100000 non-null  float64
 9   dias_atraso                100000 non-null  float64
 10  num_pagamentos_atrasados   100000 non-null  float64
 11  num_verificacoes_credito   100000 non-null  float64
 12  mix_credito                100000 non-null  object
 13  divida_total               100000 non-null  float64
 14  taxa_uso_credito           100000 non-null  float64
 15  idade_historico_credito    100000 non-null  float64
 16  investimento_mensal        100000 non-null  float64
 17  comportamento_pagamento    100000 non-null  object
 18  saldo_final_mes            100000 non-null  float64
 19  score_credito              100000 non-null  object
 20  emprestimo_carro           100000 non-null  int64
 21  emprestimo_casa            100000 non-null  int64
 22  emprestimo_pessoal         100000 non-null  int64
 23  emprestimo_credito         100000 non-null  int64
 24  emprestimo_estudantil      100000 non-null  int64
dtypes: float64(14), int64(7), object(4)
memory usage: 19.1+ MB
None
Index(['id_cliente', 'mes', 'idade', 'profissao', 'salario_anual',
       'num_contas', 'num_cartoes', 'juros_emprestimo', 'num_emprestimos',
       'dias_atraso', 'num_pagamentos_atrasados', 'num_verificacoes_credito',
       'mix_credito', 'divida_total', 'taxa_uso_credito',
       'idade_historico_credito', 'investimento_mensal',
```

```
       'comportamento_pagamento', 'saldo_final_mes', 'score_credito',
       'emprestimo_carro', 'emprestimo_casa', 'emprestimo_pessoal',
       'emprestimo_credito', 'emprestimo_estudantil'],
      dtype='object')
```

[3]:
```
from sklearn.preprocessing import LabelEncoder
# vai transformar as colunas de texto em números, ex: profissoes vai sair de␣
 ↪cientista, professor, mecanico, etc para 0, 1, 2, etc
codificador = LabelEncoder()

# só não aplicamos na coluna de score_credito que é o nosso objetivo
for coluna in tabela.columns:
    if tabela[coluna].dtype == "object" and coluna != "score_credito":
        tabela[coluna] = codificador.fit_transform(tabela[coluna])

# verificando se realmente todas as colunas foram modificadas
print(tabela.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 25 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   id_cliente                100000 non-null  int64
 1   mes                       100000 non-null  int64
 2   idade                     100000 non-null  float64
 3   profissao                 100000 non-null  int32
 4   salario_anual             100000 non-null  float64
 5   num_contas                100000 non-null  float64
 6   num_cartoes               100000 non-null  float64
 7   juros_emprestimo          100000 non-null  float64
 8   num_emprestimos           100000 non-null  float64
 9   dias_atraso               100000 non-null  float64
 10  num_pagamentos_atrasados  100000 non-null  float64
 11  num_verificacoes_credito  100000 non-null  float64
 12  mix_credito               100000 non-null  int32
 13  divida_total              100000 non-null  float64
 14  taxa_uso_credito          100000 non-null  float64
 15  idade_historico_credito   100000 non-null  float64
 16  investimento_mensal       100000 non-null  float64
 17  comportamento_pagamento   100000 non-null  int32
 18  saldo_final_mes           100000 non-null  float64
 19  score_credito             100000 non-null  object
 20  emprestimo_carro          100000 non-null  int64
 21  emprestimo_casa           100000 non-null  int64
 22  emprestimo_pessoal        100000 non-null  int64
 23  emprestimo_credito        100000 non-null  int64
 24  emprestimo_estudantil     100000 non-null  int64
```

```
dtypes: float64(14), int32(3), int64(7), object(1)
memory usage: 17.9+ MB
None
```

[4]:
```
# escolhendo quais colunas vamos usar para treinar o modelo
# y é a coluna que queremos que o modelo calcule
# x vai todas as colunas que vamos usar para prever o score de credito, não␣
 ↪vamos usar a coluna id_cliente porque ela é um numero qualquer que nao ajuda␣
 ↪a previsao
x = tabela.drop(["score_credito", "id_cliente"], axis=1)
y = tabela["score_credito"]

from sklearn.model_selection import train_test_split

# separamos os dados em treino e teste. Treino vamos dar para os modelos␣
 ↪aprenderem e teste vamos usar para ver se o modelo aprendeu corretamente
x_treino, x_teste, y_treino, y_teste = train_test_split(x, y, test_size=0.3,␣
 ↪random_state=1)
```

[5]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier

modelo_arvore = RandomForestClassifier() # modelo arvore de decisao
modelo_knn = KNeighborsClassifier() # modelo do KNN (nearest neighbors - vizinhos mais proximos)

# treinando os modelos
modelo_arvore.fit(x_treino, y_treino)
modelo_knn.fit(x_treino, y_treino)
```

[5]:
```
KNeighborsClassifier()
```

[6]:
```
# se o nosso modelo chutasse tudo "Standard", qual seria a acurácia do modelo?
contagem_scores = tabela["score_credito"].value_counts()
print(contagem_scores['Standard'] / sum(contagem_scores))
```

```
0.53174
```

[7]:
```
from sklearn.metrics import accuracy_score

# calculamos as previsoes
previsao_arvore = modelo_arvore.predict(x_teste)
previsao_knn = modelo_knn.predict(x_teste.to_numpy())

# comparamos as previsoes com o y_teste
# esse score queremos o maior (maior acuracia, mas tb tem que ser maior do que o chute de tudo Standard)
print(accuracy_score(y_teste, previsao_arvore))
```

```
print(accuracy_score(y_teste, previsao_knn))
```

```
C:\Users\root\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X
does not have valid feature names, but KNeighborsClassifier was fitted with
feature names
  warnings.warn(
```

```
0.8261
0.7324
```

[8]:
```
# fazendo novas previsões
novos_clientes = pd.read_csv("novos_clientes.csv")
display(novos_clientes)
for coluna in novos_clientes.columns:
    if novos_clientes[coluna].dtype == "object" and coluna != "score_credito":
        novos_clientes[coluna] = codificador.
 ↪fit_transform(novos_clientes[coluna])

previsoes = modelo_arvore.predict(novos_clientes)
print(previsoes)
```

```
   mes  idade   profissao  salario_anual  num_contas  num_cartoes  \
0    1   31.0  empresario      19300.340         6.0          7.0
1    4   32.0    advogado      12600.445         5.0          5.0
2    2   48.0  empresario      20787.690         8.0          6.0

   juros_emprestimo  num_emprestimos  dias_atraso  num_pagamentos_atrasados  \
0              17.0              5.0         52.0                      19.0
1              10.0              3.0         25.0                      18.0
2              14.0              7.0         24.0                      14.0

   …  taxa_uso_credito idade_historico_credito  investimento_mensal  \
0  …         29.934186                   218.0             44.50951
1  …         28.819407                    12.0              0.00000
2  …         34.235853                   215.0              0.00000

       comportamento_pagamento  saldo_final_mes  emprestimo_carro  \
0  baixo_gasto_pagamento_baixo       312.487689                 1
1  baixo_gasto_pagamento_medio       300.994163                 0
2   baixo_gasto_pagamento_alto       345.081577                 0

  emprestimo_casa  emprestimo_pessoal  emprestimo_credito  \
0               1                   0                   0
1               0                   0                   0
2               1                   0                   1

   emprestimo_estudantil
0                      0
1                      1
```

```
2                       0

[3 rows x 23 columns]
```

```
['Poor' 'Good' 'Standard']
```

[9]:
```
# quais as caracteristicas mais importantes para definir o score de credito?
colunas = list(x_teste.columns)
importancia = pd.DataFrame(index=colunas, data=modelo_arvore.
 ↪feature_importances_)
importancia = importancia * 100
print(importancia)
```

```
                                 0
mes                          4.003337
idade                        4.247354
profissao                    3.277713
salario_anual                5.122331
num_contas                   3.568979
num_cartoes                  4.621391
juros_emprestimo             7.507804
num_emprestimos              3.085498
dias_atraso                  6.568754
num_pagamentos_atrasados     4.550609
num_verificacoes_credito     4.478127
mix_credito                  8.131626
divida_total                12.007957
taxa_uso_credito             5.079802
idade_historico_credito      7.633325
investimento_mensal          4.843014
comportamento_pagamento      2.377697
saldo_final_mes              5.425015
emprestimo_carro             0.697623
emprestimo_casa              0.710199
emprestimo_pessoal           0.688423
emprestimo_credito           0.695516
emprestimo_estudantil        0.677906
```

Conclusão: Esse projeto mostra como podemos fazer a importação, tratamento de dados e análise de dados. Com isso, foi possível treinar dois modelos de classificação para auxiliar na previsão dos dados. Isso ajudaria a empresa a verificar com uma acurácia de 82% quais os clientes que possuem um bom score. Outro ponto, é que foi possível analisar quais as características mais importantes para definir esse score do cliente.

[ ]: