Projeto Python IA: Inteligência Artificial e Previsões

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

In [ ]:
```
import pandas as pd

table = pd.read_csv("clientes.csv") # read csv file
table = table.dropna() # remove rows with NaN values
display(table)
display(table.info())
```

	id_cliente	mes	idade	profissao	salario_anual	num_contas	num_cartoes	juros_
0	3392	1	23.0	cientista	19114.12	3.0	4.0	
1	3392	2	23.0	cientista	19114.12	3.0	4.0	
2	3392	3	23.0	cientista	19114.12	3.0	4.0	
3	3392	4	23.0	cientista	19114.12	3.0	4.0	
4	3392	5	23.0	cientista	19114.12	3.0	4.0	
...	...	...	...	...	...	...	...	
99995	37932	4	25.0	mecanico	39628.99	4.0	6.0	
99996	37932	5	25.0	mecanico	39628.99	4.0	6.0	
99997	37932	6	25.0	mecanico	39628.99	4.0	6.0	
99998	37932	7	25.0	mecanico	39628.99	4.0	6.0	
99999	37932	8	25.0	mecanico	39628.99	4.0	6.0	

100000 rows × 25 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
```

In [ ]:
```python
from sklearn.preprocessing import LabelEncoder # To represent string columns as
# remove string columns, because we can't use them in the model,the AI doesn't u
coder = LabelEncoder()
table['profissao'] = coder.fit_transform(table['profissao'])

table['mix_credito'] = coder.fit_transform(table['mix_credito'])

table["comportamento_pagamento"] = coder.fit_transform(table["comportamento_paga

display(table.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
```

In [ ]:
```python
Y = table["score_credito"] # the result we want to predict

X = table.drop(columns= ["id_cliente","score_credito"]) # the data we will use t

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(X,Y) # split the data into t
```

In [ ]:
```python
# Create AI model
from sklearn.ensemble import RandomForestClassifier # Random Forest Classifier
from sklearn.neighbors import KNeighborsClassifier     # KNeighbors Classifier

# Make AI

model_Forest = RandomForestClassifier()
model_KNN = KNeighborsClassifier()

# Train AI
model_Forest.fit(x_train, y_train)
model_KNN.fit(x_train, y_train)
```

Out[ ]:
```
KNeighborsClassifier()
```

In [ ]:
```
# Choose the best model
predict_Forest = model_Forest.predict(x_test)
predict_KNN = model_KNN.predict(x_test)

from sklearn.metrics import accuracy_score

display(accuracy_score(y_test, predict_Forest)) # Compare the results of the moa
display(accuracy_score(y_test, predict_KNN))
```

```
0.8296
0.73996
```

In [ ]:
```
# Use the model
new_clients = pd.read_csv("novos_clientes.csv") # read the new clients file
display(new_clients)

coder = LabelEncoder()
new_clients['profissao'] = coder.fit_transform(new_clients['profissao'])

new_clients['mix_credito'] = coder.fit_transform(new_clients['mix_credito'])

new_clients["comportamento_pagamento"] = coder.fit_transform(new_clients["compor

predict = model_Forest.predict(new_clients) # predict the new clients score
display(predict)
```

	mes	idade	profissao	salario_anual	num_contas	num_cartoes	juros_emprestimo	n
0	1	31.0	empresario	19300.340	6.0	7.0	17.0	
1	4	32.0	advogado	12600.445	5.0	5.0	10.0	
2	2	48.0	empresario	20787.690	8.0	6.0	14.0	

3 rows × 23 columns

```
array(['Poor', 'Good', 'Good'], dtype=object)
```