Avaliação de risco de crédito usando modelos de classificação (I)

A - INFORMAÇÕES INICIAIS

Objetivo do projeto:

Avaliar qual modelo de machine learning, dentre os seis abaixo, apresenta a melhor acurária na classificação de risco para concessão de novos créditos a partir de uma determinada base de clientes, usando a métrica de Acurácia como referência.

Modelos utilizados

- Árvore de decisão
- Random Forest
- Bagging
- AdaBoost
- Extremely Randomized Forest
- Gradient Boosting

Base de dados

O dataset utilizado chama-se **Default od Credit Cards** e foi disponibilizado pela UCI. Ele possui 1 coluna de identificação, 23 variáveis explicativas, 1 variável dependente binária e 30.000 observações (sem valores ausentes).

Link para o dataset: https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients

Variáveis

Target:

- Default payment / Atraso no pagamento (Sim = 1, Não = 0)

Explicativas:

- Amount of given credit / Valor do crédito concedido (em dólar)
- Gender / Gênero (1 = masculino; 2 = feminino)
- Education / Educação (1 = graduado; 2 = estudante universitário; 3 = ensino médio; 4 = outros)
- Marital status / Status marital (1 = casado; 2 = solteiro; 3 = outros)
- Age / Idade (anos)
- History of past payment / Histórico de pagamentos (Abril-Setembro/2005)
- Amount of bill statement / Valor faturado (Abril-Setembro/2005; em dólar)
- Amount of previous payment / Valor pago (Abril-Setembro/2005, em dólar)

B - IMPORTAÇÕES, ANÁLISES E DEFINIÇÃO DE VARIÁVEIS E BASES DE TREINO E TESTE

Bibliotecas e funções

In [1]:
```
import pandas as pd
import numpy as np
import sklearn as sk
import matplotlib as mpl
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import GradientBoostingClassifier

from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
```

Dataset

In [2]:
```
credit_data = pd.read_excel('credit.xls', skiprows = 1)
```

Análise exploratória

In [3]:
```
credit_data.shape
```

Out[3]:
```
(30000, 25)
```

In [4]:
```
# Visualização das variáveis e das primeiras linhas
print(credit_data.head())
```

```
   ID  LIMIT_BAL  SEX  EDUCATION  MARRIAGE  AGE  PAY_0  PAY_2  PAY_3  PAY_4  \
0   1      20000    2          2         1   24      2      2     -1     -1
1   2     120000    2          2         2   26     -1      2      0      0
2   3      90000    2          2         2   34      0      0      0      0
3   4      50000    2          2         1   37      0      0      0      0
4   5      50000    1          2         1   57     -1      0     -1      0

   ...  BILL_AMT4  BILL_AMT5  BILL_AMT6  PAY_AMT1  PAY_AMT2  PAY_AMT3  \
0  ...          0          0          0         0       689         0
1  ...       3272       3455       3261         0      1000      1000
2  ...      14331      14948      15549      1518      1500      1000
3  ...      28314      28959      29547      2000      2019      1200
4  ...      20940      19146      19131      2000     36681     10000

   PAY_AMT4  PAY_AMT5  PAY_AMT6  default payment next month
0         0         0         0                          1
1      1000         0      2000                          1
2      1000      1000      5000                          0
3      1100      1069      1000                          0
4      9000       689       679                          0
```

```
[5 rows x 25 columns]
```

Definição das variáveis e separação das bases de treinos e testes

In [5]:
```
# Variável target
target = 'default payment next month'
y = np.asarray(credit_data[target])
```

In [6]:
```
# Variáveis explicativas
features = credit_data.columns.drop(['ID', target])
X = np.asarray(credit_data[features])
```

In [7]:
```
# Bases para treino (70%) e teste (30%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_s
```

C - MODELOS DE CLASSIFICAÇÃO (VERSÃO 1)

A primeira versão de cada modelo será criada usando os hiperparâmetros pré-definidos pelo sklearn.

Árvore de Decisão

In [8]:
```
# Criação do modelo
arvore = DecisionTreeClassifier()

# Treinamento
arvore.fit(X_train, y_train)

# Predição
arvore_pred = arvore.predict(X_test)

# Matriz de confusão
print(confusion_matrix(y_test, arvore_pred))

# Acurácia
print("Acurácia = ", accuracy_score(y_test, arvore_pred))
```

```
[[5625 1357]
 [1206  812]]
Acurácia =  0.7152222222222222
```

Random Forest

In [9]:
```
# Criação do modelo
random = RandomForestClassifier()

# Treinamento
random.fit(X_train, y_train)

# Predição
random_pred = random.predict(X_test)

# Matriz de confusão
print(confusion_matrix(y_test, random_pred))
```

```
# Acurácia
print("Acurácia = ", accuracy_score(y_test, random_pred))
```

```
[[6605  377]
 [1272  746]]
Acurácia =  0.8167777777777778
```

Bagging

In [10]:

```
# Criação do modelo
bagging = BaggingClassifier()

# Treinamento
bagging.fit(X_train, y_train)

# Predição
bagging_pred = bagging.predict(X_test)

# Matriz de confusão
print(confusion_matrix(y_test, bagging_pred))

# Acurácia
print("Acurácia = ", accuracy_score(y_test, bagging_pred))
```

```
[[6553  429]
 [1319  699]]
Acurácia =  0.8057777777777778
```

AdaBoost

In [11]:

```
# Criação do modelo
ada = AdaBoostClassifier()

# Treinamento
ada.fit(X_train, y_train)

# Predição
ada_pred = ada.predict(X_test)

# Matriz de confusão
print(confusion_matrix(y_test, ada_pred))

# Acurácia
print("Acurácia = ", accuracy_score(y_test, ada_pred))
```

```
[[6715  267]
 [1364  654]]
Acurácia =  0.8187777777777778
```

Extremely Randomized Forest (Extra Trees)

In [12]:

```
# Criação do modelo
extratrees = ExtraTreesClassifier()

#Treinamento
extratrees.fit(X_train, y_train)

# Predição
extra_pred = extratrees.predict(X_test)
```

```
# Matriz de confusão
print(confusion_matrix(y_test, extra_pred))

# Acurácia
print("Acurácia = ", accuracy_score(y_test, extra_pred))
```

```
[[6579  403]
 [1290  728]]
Acurácia =  0.8118888888888889
```

Gradient Boosting

In [13]:

```
# Criação do modelo
gboost = GradientBoostingClassifier()

#Treinamento
gboost.fit(X_train, y_train)

# Predição
gboost_pred = gboost.predict(X_test)

# Matriz de confusão
print(confusion_matrix(y_test, gboost_pred))

# Acurácia
print("Acurácia = ", accuracy_score(y_test, gboost_pred))
```

```
[[6658  324]
 [1293  725]]
Acurácia =  0.8203333333333334
```

D - RANKING DA 1ª VERSÃO DOS MODELOS PARA CLASSIFICAÇÃO DE RISCO DE CRÉDITO

Usando os parâmetros e hiperparâmetros definidos, por padrão, pelo sklearn, ou seja, sem fazer nenhum tipo de ajuste (como remoção de variáveis ou tunning dos hiperparâmetros, por exemplo) o melhor resultado para essa base de dados foi apresentado pelo **Gradient Boosting**, com **82,03%** de precisão. O ranking ficou assim:

- 1º) Gradient Boosting = 82,03% de acurácia
- 2º) Extra Trees = 81,89% de acurácia
- 3º) AdaBoost = 81,88% de acurácia
- 4º) Random Forest = 81,68% de acurácia
- 5º) Bagging = 80,58% de acurácia
- 6º) Árvore de decisão = 71,52% de acurácia

O objetivo para a criação das próximas versões de classificadores será conseguir acurácia = ou > do que 90%.