**Instituto Superior de Gestão e Administração de Santarém**

**Curso PG em *Data Science***

# Estudo de Campanha Comercial a Clientes Bancários

a22210497 – Inês Luís

a22203942 – Rodrigo Antunes

a22210477 – Tiago Flor

Docente:

Professor Doutor Ricardo Vardasca

Unidade Curricular: Aprendizagem Automática (*Machine Learning*)

Santarém

Ano letivo 2022-2023

[página de verso, em branco]

**Resumo**

O presente relatório visa descrever a aplicabilidade dos métodos de aprendizagem não supervisionada e supervisionada e a criação de uma *dashboard* de uma base de dados sobre uma campanha bancária.

Foram usados métodos de classificação, para identificar clientes que subscreveram um determinado produto e também foi feita uma comparação de diferentes métodos de regressão linear sobre duas características desta base de dados. Ainda foram usadas técnicas de agrupamento e de associação para identificar diferentes conjuntos de clientes.

O objetivo deste estudo é comparar a implementação das diferentes técnicas de aprendizagem automática, e concluir acerca dos resultados e limitações sobre os modelos aplicados. A conclusão obtida é que os modelos de classificação são aqueles que melhor explicam a base de dados considerada para o projeto.

*Palavras-chave: aprendizagem automática; dados; Python; clusters.*

**ÍNDICE**

INTRODUÇÃO ..... 6

DESENVOLVIMENTO DO TEMA ..... 7

Método PCA ..... 10

Modelos de Associação ..... 12

Modelos de *Clustering* ..... 13

Modelos de Classificação ..... 18

Modelos de Regressão ..... 26

DASHBOARD – POWER BI ..... 31

CONCLUSÕES ..... 33

BIBLIOGRAFIA ..... 35

## ÍNDICE DE FIGURAS

Figura 1. Importação de dados ..... 7
Figura 2. Resumo do *dataset* importado ..... 7
Figura 3. Distribuição dos clientes por estado civil ..... 8
Figura 4. Distribuição dos clientes por histórico de *default* ..... 8
Figura 5. *Boxplot* das idades da carteira de clientes ..... 9
Figura 6. Eliminação de colunas e substituição de atributos *string* para numérico ..... 9
Figura 7. *Scree plot* ..... 10
Figura 8. PCA *Biplot* ..... 12
Figura 9. Associação apriori ..... 12
Figura 10. *Clustering*: KMEANS ..... 13
Figura 11. Método do Cotovelo ..... 14
Figura 12. *Clustering: K-Means* ..... 14
Figura 13. K-*Means Clustering* ..... 15
Figura 14. *Clustering: Hierarchical* ..... 16
Figura 15. *Hierarchical Clustering* e Dendrogram ..... 16
Figura 16. *Clustering*: DBSCAN ..... 17
Figura 17. DBSCAN *Clustering* ..... 18
Figura 18. Normalização de dados ..... 18
Figura 19. Separação entre *sets* teste e treino ..... 19
Figura 20. Gráfico *k-distance* ..... 22
Figura 21. Avaliação dos modelos de classificação ..... 25
Figura 22. *Heatmap* de correlações ..... 26
Figura 23. *Scatterplot* (Previous/Pdays) ..... 27
Figura 24. Regressão Linear ..... 27
Figura 25. *Plot* Regressão Linear Simples ..... 28
Figura 26. Comparação modelos de regressão ..... 30
Figura 27. *Dashboard* – Painel 1 ..... 31
Figura 28. *Dashboard* – Painel 2 ..... 32

## ÍNDICE DE TABELAS

Tabela 1. Coeficientes Componentes PCA ..... 11
Tabela 2. Decision Tree Report ..... 20
Tabela 3. k-Nearest Neighbour Report (K = 1) ..... 21
Tabela 4. k-Nearest Neighbour Report (K = 3) ..... 21
Tabela 5. k-Nearest Neighbour Report (K = 5) ..... 22
Tabela 6. *Logistic Regression* ..... 23
Tabela 7. *Random Forest* ..... 23
Tabela 8. *ADABOOST* ..... 24

## INTRODUÇÃO

O atual estudo é realizado no âmbito da unidade curricular de Aprendizagem Automática.

O objetivo principal é aplicar os métodos de aprendizagem automática numa base de dados sobre a adesão a uma campanha bancária e que contém informações dos clientes bancários.

A aprendizagem automática é um ramo da inteligência artificial que consiste no tratamento de grandes dados de informação, através de métricas e parâmetros pré-estabelecidos, para facilitar a análise e apoiar a tomada de decisão das organizações. O tratamento da informação é realizado através de vários métodos, que permite reconhecer padrões, fazer previsões, corrigir erros, de forma acelerada, possibilitando economizar tempo e otimizar resultados. A aprendizagem automática pode ser supervisionada ou não supervisionada.

A aprendizagem supervisionada consiste na intervenção humana, isto é, o modelo é construído através de um conjunto de dados, que são rotulados pelo operador humano. São vários os métodos de aprendizagem supervisionada como regressão linear, regressão logística, árvores de decisão, redes neurais, entre outros. Por outro lado, na aprendizagem não supervisionada, não há rótulos determinados pelo humano. O algoritmo determina padrões, correlações e relacionamento através da base de dados. Existem vários algoritmos de aprendizagem não supervisionada, como análise de componentes principais (PCA), k-means, regra de associação apriori, agrupamento hierárquico, classificação Naive Bayes, entre outros.

Para este estudo, foi utilizado um *dataset* sobre uma campanha bancária, realizada através de chamadas telefônicas. Após as chamadas, cada cliente, deveria informar o banco a sua intenção de aderir, ou não, à campanha, e assim sendo, o resultado deste estudo é uma variável binária: o cliente subscreveu a campanha (sim); se o cliente não subscreveu a campanha (não).

Na primeira parte do relatório é apresentada a caracterização do conjunto de dados bem como a aplicação dos métodos de aprendizagem automática em *Python*. De seguida é apresentado *dashboard* realizada em PowerBI, seguindo-se uma breve discussão sobre os resultados obtidos.

## DESENVOLVIMENTO DO TEMA

Conforme mencionado na introdução do presente documento, o âmbito do projeto visa a análise em *Python* de um *dataset* pré-determinado, a partir de uma *pool* de *datasets* disponíveis.

Para o devido efeito, o grupo de trabalho optou pela execução do *dataset* "02". Este *dataset* representa o conjunto de resultados obtidos no decorrer de uma campanha de uma determinada instituição bancária, que visa aferir, a partir da carteira de clientes contactados, se os mesmos aderiram, ou não, ao produto de crédito que a instituição pretendia colocar.

O conjunto de dados discrimina atributos individuais de cada cliente alvo de contacto pela entidade, nomeadamente, idade, nível de educação, saldos médios na conta, situação de incumprimento, entre outros, assim como dados relativamente ao tempo de contacto e sucesso da campanha em cada cliente individual.

De forma a iniciar a análise em *Python* dos referidos dados, em primeiro lugar, foi necessário proceder à importação do *dataset*:

```
df = pd.read_csv(r"C:\Users\[redacted]\Documents\Programming\Datasets\02.csv", sep=";",
                 low_memory=False)
print(df.head())
```

Figura 1. Importação de dados

O resumo dos dados importados apresenta-se de seguinte forma:

| age | job | marital | education | ... | pdays | previous | poutcome | y |
|---|---|---|---|---|---|---|---|---|
| 58 | management | married | tertiary | ... | -1 | 0 | unknown | no |
| 44 | technician | single | secondary | ... | -1 | 0 | unknown | no |
| 33 | entrepreneur | married | secondary | ... | -1 | 0 | unknown | no |
| 47 | blue-collar | married | unknown | ... | -1 | 0 | unknown | no |
| 33 | unknown | single | unknown | ... | -1 | 0 | unknown | no |

Figura 2. Resumo do *dataset* importado

Após a importação dos dados, realizou-se uma breve análise estatística de forma a caracterizar a carteira de clientes que originou a análise:

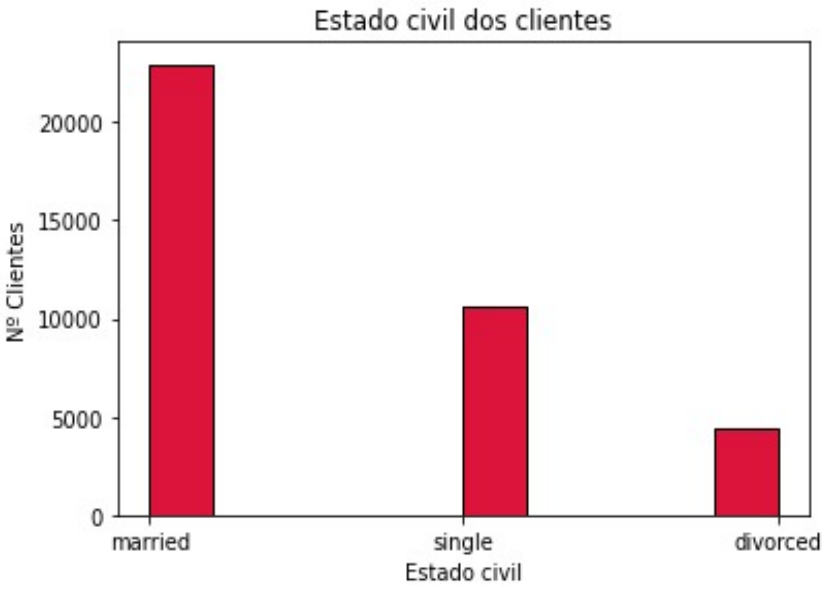


Figura 3. Distribuição dos clientes por estado civil

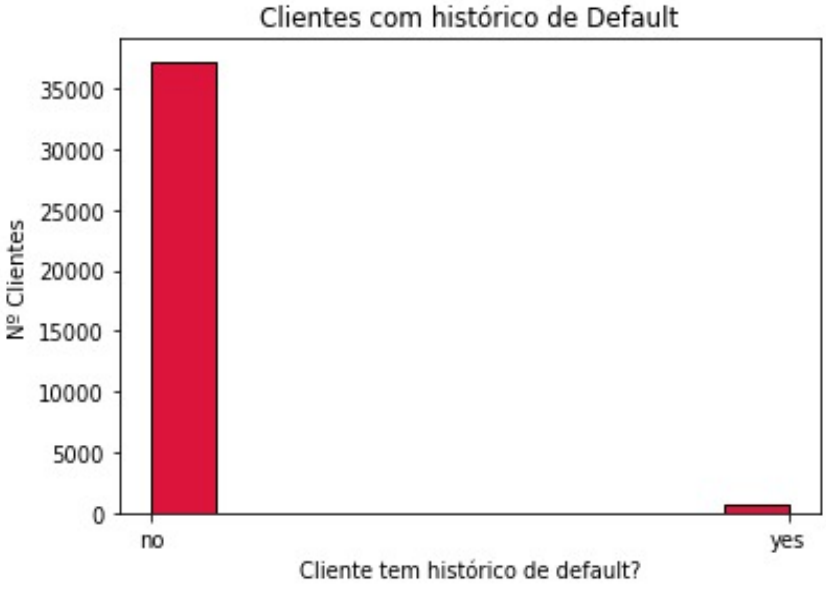


Figura 4. Distribuição dos clientes por histórico de *default*

De seguida, com a inclusão de visualização através de *pairplot* (biblioteca *seaborn*), assim como aplicação do código "df.describe()" e *box plot* (plt.boxplot(df['age'])), verifica-se existência de *outliers* nos dados.

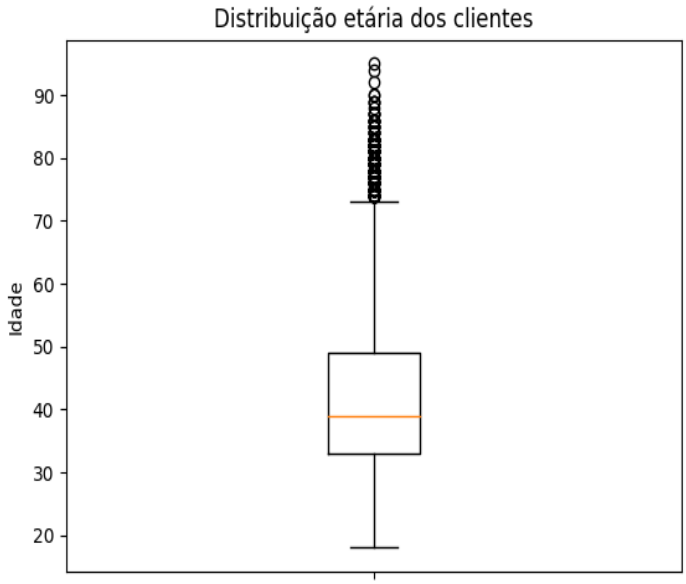


Figura 5. *Boxplot* das idades da carteira de clientes

Face à existência de *outliers* e atendendo à dimensão elevada das entradas no *dataset*, entendeu-se a aplicação do método de eliminação destes valores. Para o efeito, foi delimitado um limite inferior de valores até 15% e um limite superior desde 85%, sendo que as ocorrências fora destes limites foram excluídas. Este método resultou na eliminação de 7.241 entradas, de um total de 45.209.

Adicionalmente, foram detetados atributos categóricos como, por exemplo, a coluna "*education*", que apresenta 4 possibilidades, em formato *string*. Como forma de tratamento destes atributos, foi atribuído um classificador numérico (0, 1, 2, 3,...) em todas as colunas que apresentavam esta característica. Detetou-se, ainda, colunas contendo atributos que não foram considerados relevantes para a análise, pelo que foram eliminadas.

```
df = df.drop(['job', 'contact', 'day', 'month'], axis=1).replace(
    to_replace=['single', 'married', 'divorced', 'unknown', 'primary', 'secondary', 'tertiary', 'no', 'yes',
                'failure', 'other', 'success', 'unknown'],
    value=['1', '2', '3', '0', '1', '2', '3', '0', '1', '0', '1', '2', '3'])
```

Figura 6. Eliminação de colunas e substituição de atributos *string* para numérico

Desta forma, considerou-se o *dataframe* (df) resultante do tratamento aplicado como a base para a aplicação dos métodos de *Machine Learning*.

**Método PCA**

Usando a análise de componentes principais (PCA) conseguimos identificar padrões no conjunto de dados, transformando um conjunto de variáveis correlacionadas num conjunto de variáveis novas não correlacionadas, chamados componentes principais.

Para esta análise, primeiramente, efetuou-se uma normalização dos dados para que estes tivessem todos a mesma escala e só posteriormente executamos a análise PCA para identificar os componentes principais. Para fazer esta identificação foi calculada a variância para todos os componentes e colocado os valores num gráfico para ajudar a selecionar os componentes que conseguem explicar a maior variância nos dados. (Figura 7)

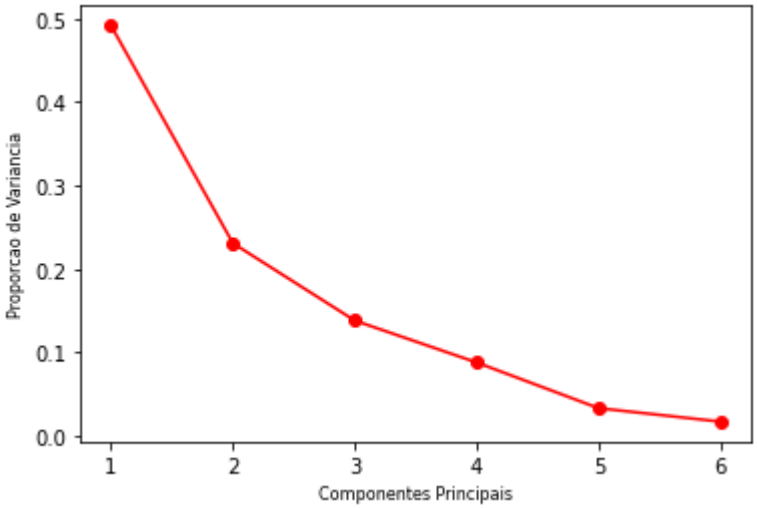


Figura 7. *Scree plot*

Tabela 1. Coeficientes Componentes PCA

| **Componente** | **Coeficientes** |
|---|---|
| 1 | 0,49 |
| 2 | 0,23 |
| 3 | 0,14 |
| 4 | 0,09 |
| 5 | 0,03 |
| 6 | 0,02 |

Através do gráfico e da tabela escolhemos os dois primeiros componentes uma vez que estes explicam aproximadamente 72% da variância total dos dados.

Projetando estes dados nos dois componentes principais obtemos o gráfico em seguida onde se pode visualizar como cada variável contribui para cada componente, PC1 e PC2 (Figura 8).

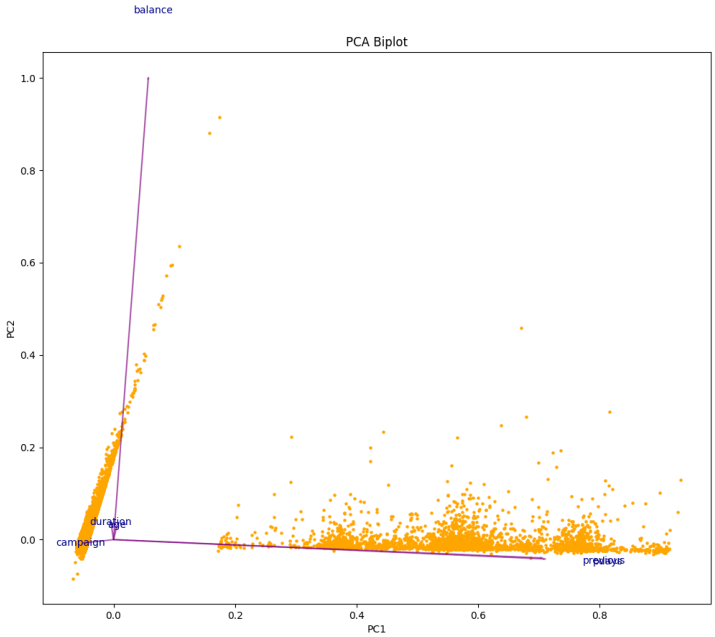


Figura 8. PCA *Biplot*

## Modelos de Associação

Para concluirmos sobre a associação do conjunto de dados do *dataset*, utilizamos o método de APRIORI (regra de associação). Este método permite identificar relações subjacentes entre diferentes itens, bem como, identificar padrões e associações frequentes entre conjunto de itens.

```
# Nova importação do dataset, com drop de colunas com valores não categóricos
df3 = pd.read_csv("02.csv", sep=";",
                  low_memory=False, index_col=False).drop(
    ['age', 'default', 'balance', 'housing', 'loan', 'day', 'duration', 'campaign', 'pdays', 'previous'], axis=1)

# Retirar uma amostra aleatória com 'N' linhas para reduzir recursos de computação.
N = 43000
df_apy = df3.drop(df3.sample(N).index)
print(df_apy)

transactions = []
size = len(df_apy)
cols = len(df_apy.columns)
for i in range(0, size):
    transactions.append([str(df_apy.values[i, j])
                         for j in range(0, cols)])

rules = apriori(transactions=transactions, min_support=0.1,
                min_confidence=0.95, min_lift=1, min_length=3)
results = list(rules)
print(results)
```

Figura 9. Associação apriori

Para esta análise excluímos da base de dados as colunas *'job'*, *'contact'*, *'day'*, *'month'*, *'pdays'*, *'balance'*, *'duration'*. Para uma melhor precisão, foi excluído da análise uma amostra aleatória de 43000 linhas. Com recurso à biblioteca *mlxtend*, para calcular o algoritmo apriori, com uma confiança de 95%,

**Modelos de *Clustering***

Agrupamento - KMEANS

O KMEANS é um método de agrupamento, que divide os dados em grupos não sobrepostos, ou seja, agrupa um conjunto de dados num conjunto de observações em k *clusters*.

```
x = df
wcss = []
for i in range(1, 7):
    kmeans = KMeans(i)
    kmeans.fit(x)
    wcss_iter = kmeans.inertia_
    wcss.append(wcss_iter)
# Plot 'método do cotovelo' para determinar número de clusters
number_clusters = range(1, 7)
plt.plot(number_clusters, wcss)
plt.title('Metodo do cotovelo')
plt.xlabel('Numero de clusters')
plt.ylabel('WCSS')
plt.show()
```

Figura 10. *Clustering*: KMEANS

Inicialmente, para determinar o número ideal de agrupamentos, foi utilizada a técnica do cotovelo, concluindo que o número ideal de agrupamentos pode ser três, como pode ser percetível na Figura 11, onde, a linha (WCSS) apresenta uma inclinação mais acentuada no ponto três.

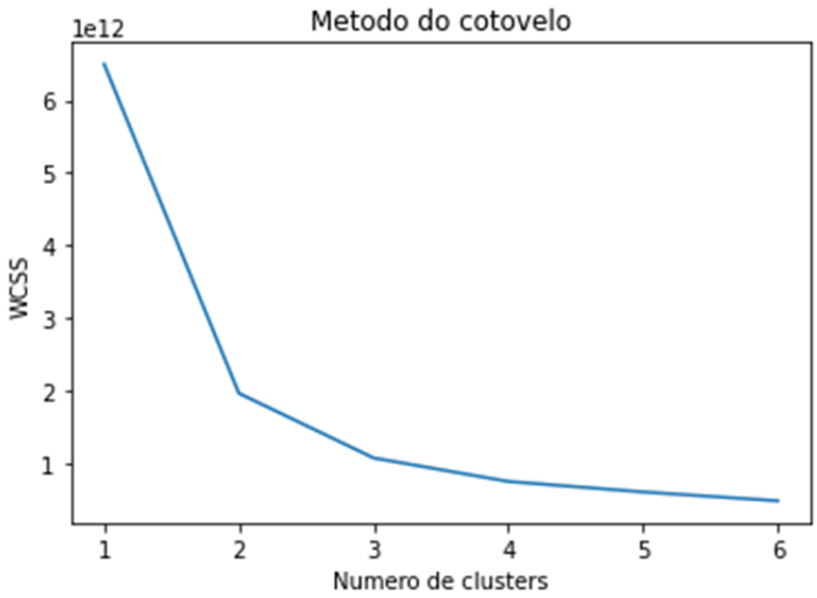


Figura 11. Método do Cotovelo

De seguida, utilizando a biblioteca *scikit-learn* definiu-se o algoritmo com três *clusters*, e elaborou-se um gráfico de dispersão para relacionar as variáveis '*duration'* (eixo do x) e '*age'* (eixo do y).

```
# Kmeans clustering
k_means = KMeans(n_clusters=3, random_state=42)
k_means.fit(x)
x['KMeans_labels'] = k_means.labels_

identified_clusters = k_means.fit_predict(x)
print('clusters identificados', identified_clusters)
data_with_clusters = x.copy()
data_with_clusters['Clusters'] = identified_clusters

# Scatter plot com os clusters identificados
colors = ['purple', 'red', 'blue', 'green']
plt.figure(figsize=(10, 10))
plt.scatter(data_with_clusters['duration'], data_with_clusters['age'], c=data_with_clusters['Clusters']
            cmap=matplotlib.colors.ListedColormap(colors), s=15)
plt.title('K-Means Clustering', fontsize=20)
plt.xlabel('Duration', fontsize=14)
plt.ylabel('Age', fontsize=14)
plt.show()
```

Figura 12. *Clustering: K-Means*

A Figura 13, representa o gráfico de dispersão obtido, com recurso à biblioteca *matplotlib*. A cor de cada ponto determina o grupo de cluster a que pertence.

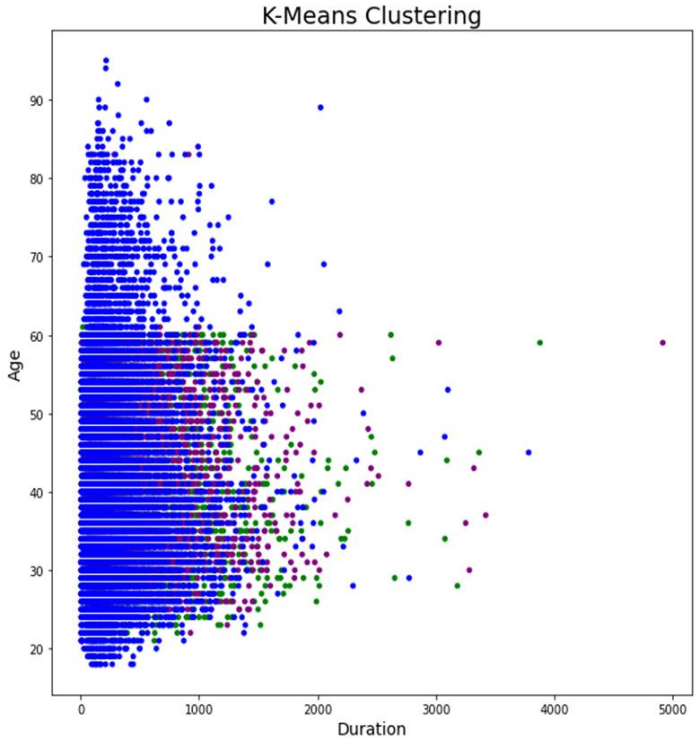


Figura 13. K-*Means Clustering*

Agrupamento - Hierárquico

O agrupamento hierárquico permite construir uma hierarquia de *clusters*. Existem dois tipos de agrupamentos hierárquicos: agrupamento aglomerativo – começa por misturar os clusters que são mais semelhantes, até que fiquem todos agrupados num único cluster – e agrupamento divisivo – os clusters menos semelhantes são separados.

Para visualizar a estrutura do agrupamento de dados usando o agrupamento de dados hierárquico, foram utilizadas as variáveis '*age'* e '*balance'* para efetuar um gráfico de dispersão para visualizar os *clusters*. Foi ainda elaborado um dendrograma, com recurso à biblioteca *scipy*.

```
# 2. _____HIERÁRQUICO_____
model = AgglomerativeClustering(n_clusters=3, affinity='euclidean')
model.fit(x)

x['HR_labels'] = model.labels_

# Plotting resulting clusters
plt.figure(figsize=(10, 10))
plt.scatter(data_with_clusters['balance'], data_with_clusters['age'], c=x['HR_labels'],
            cmap=matplotlib.colors.ListedColormap(colors), s=15)
plt.title('Hierarchical Clustering', fontsize=20)
plt.xlabel('Balance', fontsize=14)
plt.ylabel('Age', fontsize=14)
plt.show()

selected_data = df.iloc[1000:1500, [1,5]]
clusters = shc.linkage(selected_data,
            method='ward',
            metric="euclidean")
shc.dendrogram(Z=clusters)
plt.show()
```

Figura 14. *Clustering: Hierarchical*

As relações hierárquicas são evidenciadas nos gráficos seguintes (Figura 15). No gráfico de dispersão, o agrupamento hierárquico de clusters é definido pelas cores, enquanto que, no dendrograma, o agrupamento é identificado pelo formato (formato de árvore) que é originado pela distância.

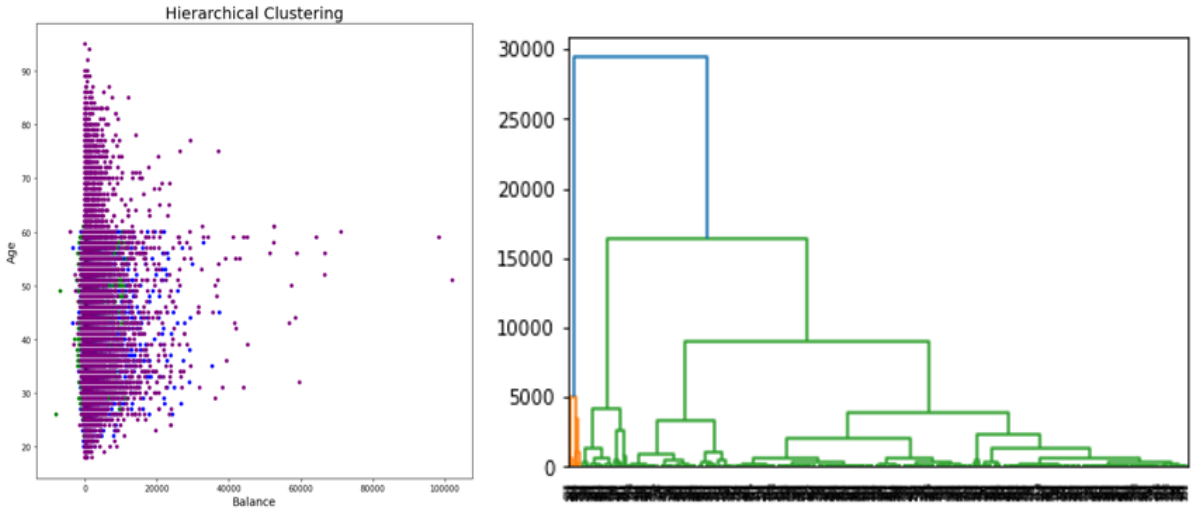


Figura 15. *Hierarchical Clustering* e Dendrogram

Agrupamento - DBSCAN

O agrupamento DBSCAN (*Density-based spatial clustering of applications with noise*) é um método de agrupamento que é definido tendo em conta a densidade.

Para efetuar esta análise foram utilizadas as variáveis '*balance'* e '*age'*, sendo que, o algoritmo é definido por três parâmetros: a distância máxima entre dois pontos é igual a 250, o mínimo de pontos para se formar um cluster é 2, e a medida de distância utilizada é a *manhattan*.

```
# 3. _____DBSCAN_____
dbscan = DBSCAN(eps=250, min_samples=2, metric='manhattan')
dbscan.fit(x)

x['DBSCAN_labels'] = dbscan.labels_

# Plotting resulting clusters
plt.figure(figsize=(10, 10))
plt.scatter(data_with_clusters['balance'], data_with_clusters['age'], c=x['DBSCAN_labels'],
            cmap=matplotlib.colors.ListedColormap(colors), s=15)
plt.title('DBSCAN Clustering', fontsize=20)
plt.xlabel('Feature 1', fontsize=14)
plt.ylabel('Feature 2', fontsize=14)
plt.show()
```

Figura 16. *Clustering*: DBSCAN

A Figura 17 representa o gráfico de dispersão onde pode ser observado o agrupamento dos *clusters* através das cores.

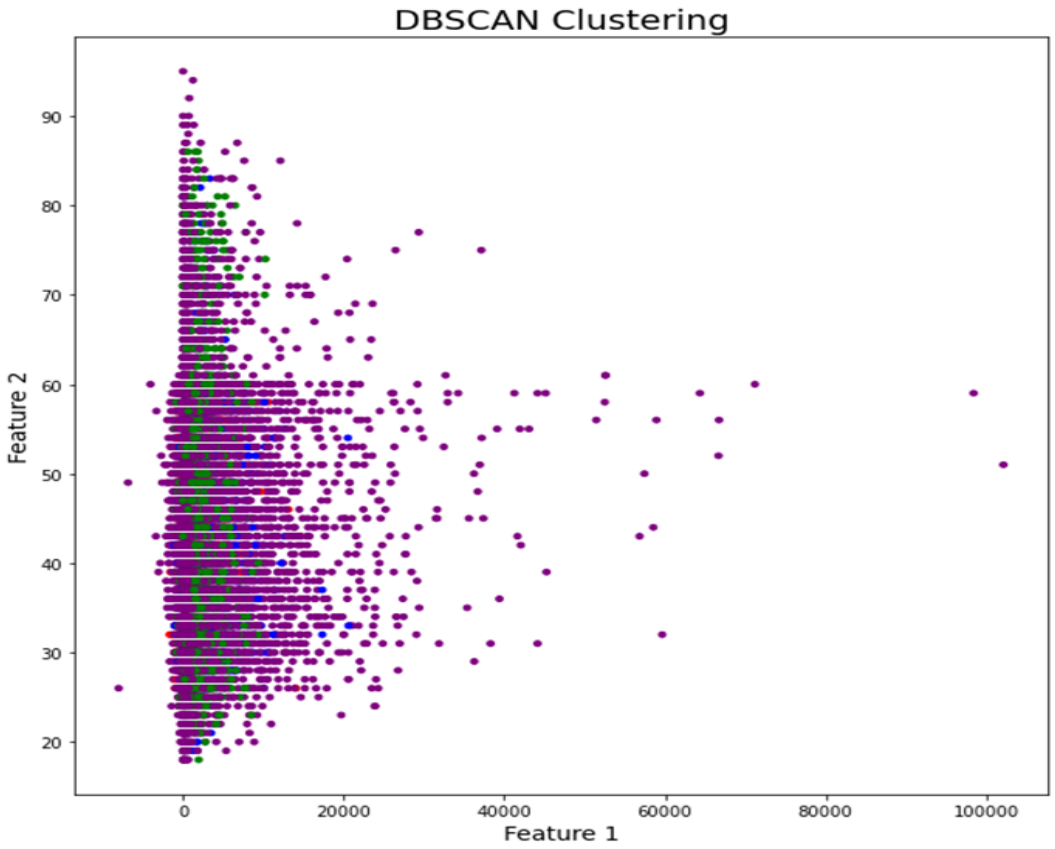


Figura 17. DBSCAN *Clustering*

**Modelos de Classificação**

Relativamente à análise do *dataset* tomando como base os modelos de classificação, foram considerados 7 (sete) modelos regularmente utilizados em *machine learning*, com objetivo de, após a validação de cada modelo, concluir qual melhor se adapta ao conjunto de dados apresentado.

Numa fase inicial, foi necessário efetuar a leitura dos dados, tendo-se observado a existência de valores negativos nas colunas "balance" e "pdays" do *dataset*, impossibilitando a classificação com base em determinados modelos, como, por exemplo, Naive-Bayes Multinomial. Face a este impedimento, optou-se por realizar *a priori* a normalização dos valores destes atributos:

```
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(df.loc[:, ['balance']])
df['balance'] = pd.DataFrame(x_scaled)  # Normalização dos valores da coluna 'balance'
x_scaled = min_max_scaler.fit_transform(df.loc[:, ['pdays']])
df['pdays'] = pd.DataFrame(x_scaled)  # Normalização dos valores da coluna 'pdays'
```

Figura 18. Normalização de dados

Concluindo-se a normalização de dados, realizou-se a separação do *dataset* entre *sets* de teste e de treino, a utilizar posteriormente em cada modelo de classificação, tendo-se optado por uma divisão 70/30:

```
X = df.loc[:, ['age', 'poutcome', 'duration', 'pdays', 'balance', 'campaign']]
y = df.iloc[:, -1].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=0)
```

Figura 19. Separação entre *sets* teste e treino

Adicionalmente, face ao risco de *overfitting* e *underfitting* dos dados, aplicou-se a validação cruzada.

1. Naive-Bayes (NB)

Previamente à aplicação deste modelo de classificação, realizou-se um processo de estandardização dos dados através do módulo *StandardScaler*, presente na biblioteca *Scikit-Learn*. Foram considerados 3 (três) métodos de eliminação, descritos infra, tendo-se apurado o respetivo grau de precisão e a matriz de confusão.

**a. Eliminação Gaussiana**

Gaussian NB

Accuracy: 86.24 %

Matriz confusão:

[[9405 811]

[ 756 419]]

**b. Multinomial**

Multinomial NB

Accuracy: 79.42 %

Matriz confusão:

[[8336 1880]

[ 464 711]]

### c. Bernoulli

Bernoulli NB

Accuracy: 89.68 %

Matriz confusão:

[[10216 0]

[ 1175 0]]

2. Árvores de Decisão

Tabela 2. Decision Tree Report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.93 | 0.92 | 0.92 | 10216 |
| 1 | 0.33 | 0.37 | 0.35 | 1175 |
| **accuracy** | | | 0.86 | 11391 |
| **macro avg** | 0.63 | 0.64 | 0.63 | 11391 |
| **weighted avg** | 0.87 | 0.86 | 0.86 | 11391 |

Confusion matrix:

[[9352 864]

[[ 744 431]]

3. k-Nearest Neighbour

a. K1

Tabela 3. k-Nearest Neighbour Report (K = 1)

| | **precision** | **recall** | **f1-score** | **support** |
|---|---|---|---|---|
| 0 | 0.92 | 0.92 | 0.92 | 10216 |
| 1 | 0.30 | 0.30 | 0.30 | 1175 |
| **accuracy** | | | 0.86 | 11391 |
| **macro avg** | 0.61 | 0.61 | 0.61 | 11391 |
| **weighted avg** | 0.86 | 0.86 | 0.86 | 11391 |

Confusion matrix:

[[9414 802]

[[ 824 351]]

b. K3

Tabela 4. k-Nearest Neighbour Report (K = 3)

| | **precision** | **recall** | **f1-score** | **support** |
|---|---|---|---|---|
| 0 | 0.92 | 0.96 | 0.94 | 10216 |
| 1 | 0.41 | 0.26 | 0.32 | 1175 |
| **accuracy** | | | 0.89 | 11391 |
| **macro avg** | 0.66 | 0.61 | 0.63 | 11391 |
| **weighted avg** | 0.87 | 0.89 | 0.87 | 11391 |

Confusion matrix:

[[9773 443]

[ 866 309]]

c. K5

Tabela 5. k-Nearest Neighbour Report (K = 5)

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.92 | 0.96 | 0.94 | 10216 |
| 1 | 0.45 | 0.25 | 0.32 | 1175 |
| **accuracy** | | | 0.89 | 11391 |
| **macro avg** | 0.68 | 0.61 | 0.63 | 11391 |
| **weighted avg** | 0.87 | 0.89 | 0.88 | 11391 |

Confusion matrix:

[[9855 361]

[ 885 290]]

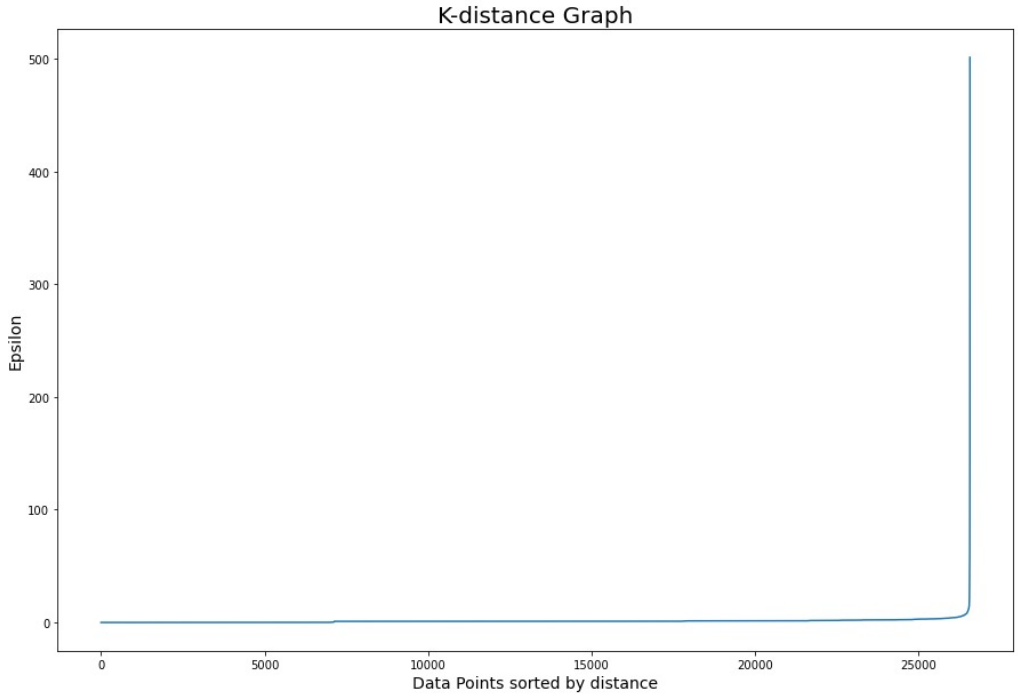


Figura 20. Gráfico *k-distance*

4. Regressão Logística

Tabela 6. *Logistic Regression*

| | **precision** | **recall** | **f1-score** | **support** |
|---|---|---|---|---|
| 0 | 0.91 | 0.98 | 0.95 | 10216 |
| 1 | 0.57 | 0.20 | 0.29 | 1175 |
| **accuracy** | | | 0.90 | 11391 |
| **macro avg** | 0.74 | 0.59 | 0.62 | 11391 |
| **weighted avg** | 0.88 | 0.90 | 0.88 | 11391 |

Confusion matrix:

[[10039 177]

[ 943 232]]

5. Random Forest

Tabela 7. *Random Forest*

| | **precision** | **recall** | **f1-score** | **support** |
|---|---|---|---|---|
| 0 | 0.91 | 0.99 | 0.95 | 10216 |
| 1 | 0.67 | 0.10 | 0.17 | 1175 |
| **accuracy** | | | 0.90 | 11391 |
| **macro avg** | 0.79 | 0.55 | 0.56 | 11391 |
| **weighted avg** | 0.88 | 0.90 | 0.87 | 11391 |

Confusion matrix:

[[10159 57]

[ 1057 118]]

6. Adaboost

Tabela 8. *ADABOOST*

| | precision | recall | f1-score | suppor |
|---|---|---|---|---|
| 0 | 0.92 | 0.97 | 0.95 | 10216 |
| 1 | 0.54 | 0.28 | 0.37 | 1175 |
| **accuracy** | | | 0.90 | 11391 |
| **macro avg** | 0.73 | 0.63 | 0.66 | 11391 |
| **weighted avg** | 0.88 | 0.90 | 0.89 | 11391 |

Confusion matrix:

[[9936 280]

[ 845 330]]

7. Support Vector Machines (SVM)

   a. **Kernel Poly**

      Accuracy: 89.69 %

      Matriz confusão:

      [[10188 28]

      [ 1146 29]]

   b. **Kernel RBF**

      Accuracy: 90.06 %

      Matriz confusão:

      [[10044 172]

      [ 960 215]]

### c. Kernel Sigmoid

Accuracy: 80.95 %

Matriz confusão:

[[9137 1079]

[1091 84]]

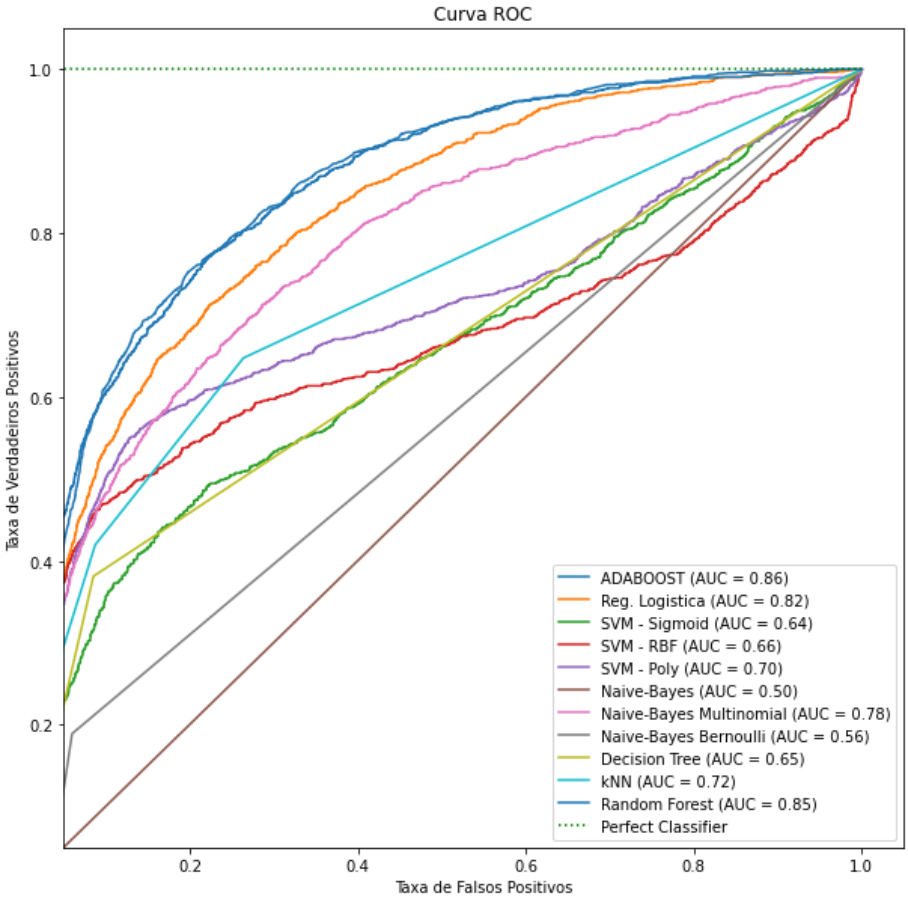


Figura 21. Avaliação dos modelos de classificação

Uma análise à aplicação dos métodos de classificação apresentados, permite constatar que foi possível atingir graus de precisão situados na escala de 80% até 90%, evidenciando-se melhor *scoring* nos modelos NB (*Bernoulli*), Regressão Logística, *ADABOOST* e SVM (RBF).

Não obstante, o elevado grau de precisão obtido nos modelos supramencionados, significa que estes também apresentam número superior de falsos negativos, comparando-os com modelos de precisão inferior.

Em termos de precisão *versus* equilíbrio entre verdadeiros positivos e negativos, destacam-se os modelos *ADABOOST*, Árvore de Decisão e NB (*Gaussian*), uma vez que, todos estes apresentam uma precisão superior a 86% e elevado rácio de verdadeiros negativos e positivos. Esta observação verifica-se após consulta do gráfico supra (Figura 21), onde estes modelos apresentam curva ROC mais aproximada de 1.0 face aos restantes.

**Modelos de Regressão**

Usando a biblioteca do *Scikit-learn* fizemos a aplicação de uma regressão linear, regressão polinomial, regressão linear Lasso, regressão Lasso CV e regressão linear *Ridge*.

Para iniciar a aplicação das regressões fizemos uma análise de correlação entre as diferentes variáveis (Figura 22), tendo optado pelas duas variáveis com a correlação maior, 'pdays' e 'previous'. (Figura 23)

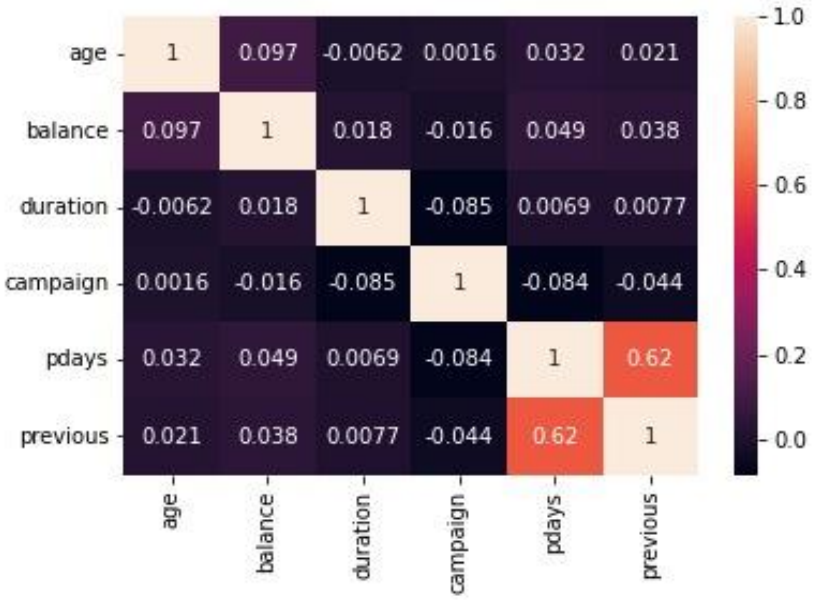


Figura 22. *Heatmap* de correlações

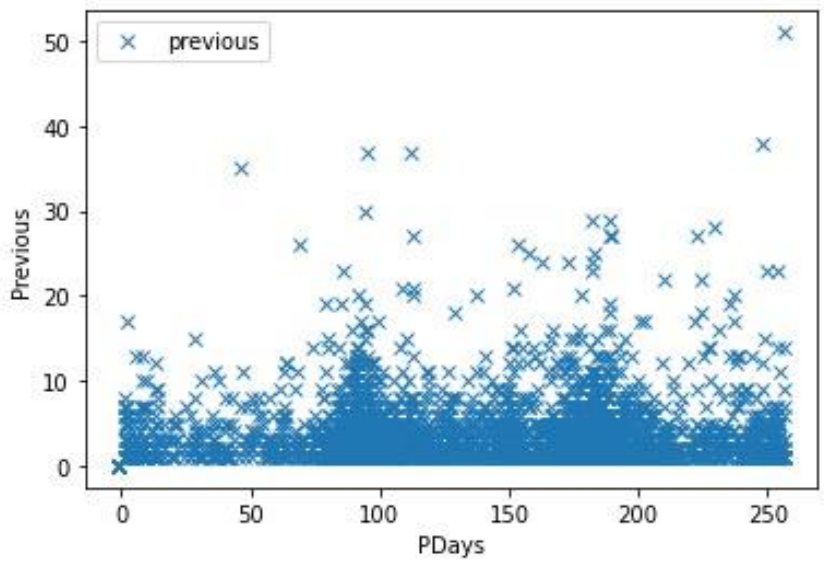


Figura 23. *Scatterplot* (Previous/Pdays)

De seguida, fizemos a divisão dos dados em conjunto de treino e em conjunto de teste e aplicamos a regressão linear simples (Figura 24), a regressão polinomial usando o método '.LinearRegression ()', a regressão linear Lasso e a regressão linear Lasso CV usando o método '.Lasso ()' e '.Lasso CV ()' respectivamente, por fim, para a aplicação da regressão linear Ridge foi usado o método 'make_pipeline()' da biblioteca 'Scikit-learn' para agilizar o processo de normalização dos dados, que é necessário realizar para posteriormente então aplicar o método '.Ridge()'.

Os resultados podem ser observados de seguida juntamente com as figuras que contêm o código utilizado.

```
# Definir colunas para a reg.Linear
y = pd.DataFrame(df['previous'])
X = pd.DataFrame(df['pdays'].values)

# Dados de Treino e Teste
X_train, X_test, y_train, y_test = train_test_split(
    X, y, train_size=0.7, test_size=0.3, random_state=1)

# Regressão Linear ----------------------------------------------------
lin_reg = LinearRegression()
lin_reg.fit(X, y)
```

Figura 24. Regressão Linear

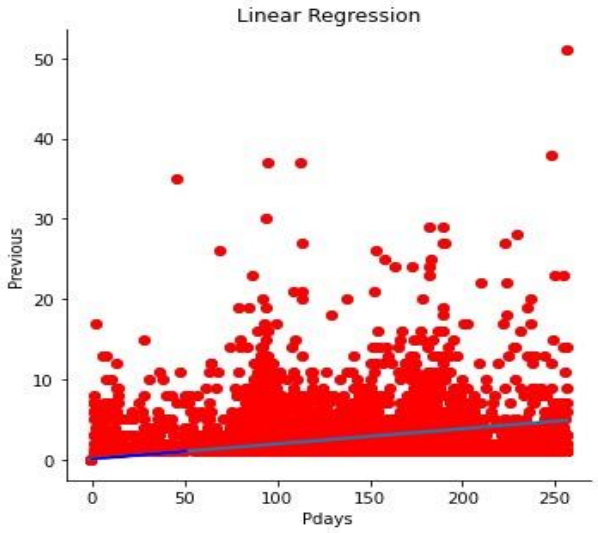


Figura 25. *Plot* Regressão Linear Simples

---
Regressão Linear Simples
Mean Absolute Error: 0.32
Mean Squared Error: 1.49
Root Mean Squared Error: 1.22
R-Squared:  0.38
Coeficiente é: [[0.01882357]]
- Validação cruzada é [0.   0.   0.43 0.31 0.31]
- A média da validação cruzada é 0.21 com um desvio-padrão de 0.18

```
# Regressão Polinomial -----------------------------------------------------
poly_regr = PolynomialFeatures(degree = 6)
X_poly_train = poly_regr.fit_transform(X_train)
X_poly_test = poly_regr.transform(X_test)
lin_reg_2 = LinearRegression()
lin_reg_2.fit(X_poly_train, y_train)

poly_reg_pred = lin_reg_2.predict(X_poly_test)
```

---
Regressão Linear Polinomial
Mean Absolute Error: 0.26
Mean Squared Error: 1.3
Root Mean Squared Error: 1.14
R-Squared:  0.46
Coeficiente são: [[ 0.00000000e+00  2.94146507e-01 -8.02036623e-03  9.83483151e-05
  -6.10280176e-07  1.86519294e-09 -2.22461878e-12]]
- Validação cruzada é [0.   0.   0.43 0.31 0.31]
- A média da validação cruzada é 0.21 com um desvio-padrão de 0.18

```
# Regressão Linear Lasso ----------------------------------------------------
lasso = Lasso(alpha=7)
lasso.fit(X_train, y_train)
lasso_pred = lasso.predict(X_test)
```

---
Regressão Linear Lasso
Mean Absolute Error: 0.35
Mean Squared Error: 1.5
Root Mean Squared Error: 1.23
R-Squared:  0.37
Coeficiente é: [0.01627816]
- Validação cruzada é [0.   0.   0.42 0.32 0.27]
- A média da validação cruzada é 0.2 com um desvio-padrão de 0.17

```
# Regressão Linear Lasso CV-------------------------------------------------
lasso_cv = LassoCV(cv=5)
lasso_cv.fit(X_train, y_train)
lasso_cv_pred = lasso_cv.predict(X_test)
```

---
Regressão Linear Lasso CV
Mean Absolute Error: 0.32
Mean Squared Error: 1.49
Root Mean Squared Error: 1.22
R-Squared:  0.38
Coeficiente é: [0.01898715]
- Validação cruzada é [0.   0.   0.43 0.31 0.31]
- A média da validação cruzada é 0.21 com um desvio-padrão de 0.18

```
# Regressão Linear Ridge ----------------------------------------------------
pipeline = make_pipeline(StandardScaler(), Ridge(alpha=1.0))
pipeline.fit(X_train, y_train)
ridge_pred = pipeline.predict(X_test)
```

---
Regressão Linear Ridge
Mean Absolute Error: 0.32
Mean Squared Error: 1.49
Root Mean Squared Error: 1.22
R-Squared:  0.38
Coeficiente é: [[0.96273562]]
- Validação cruzada é [0.   0.   0.43 0.31 0.31]
- A média da validação cruzada é 0.21 com um desvio-padrão de 0.18

Para avaliar o desempenho dos modelos foi utilizado as seguintes métricas,

- Erro absoluto médio (MAE) – média das diferenças absolutas entre as previsões e os valores reais. Quanto menor o valor melhor o modelo de regressão. (TDH, 2022)

- Erro quadrático médio (MSE) – média das diferenças quadráticas entre as previsões e os valores reais. Quanto menor o valor melhor o modelo de regressão. (TDH, 2022)

- Raiz do erro quadrático médio (RMSE) – é a raiz quadrada do MSE. Quanto menor o valor melhor o modelo de regressão. (TDH, 2022)

- (R-Squared) – É a percentagem de variabilidade da variável dependente que é explicada pela regressão. Varia de 0 a 1, sendo que quanto mais próximo de 1 melhor é o modelo. (TDH, 2022)

No final é apresentado um gráfico onde podemos observar os resultados destas métricas e comparar os valores entre os métodos utilizados (Figura 26).

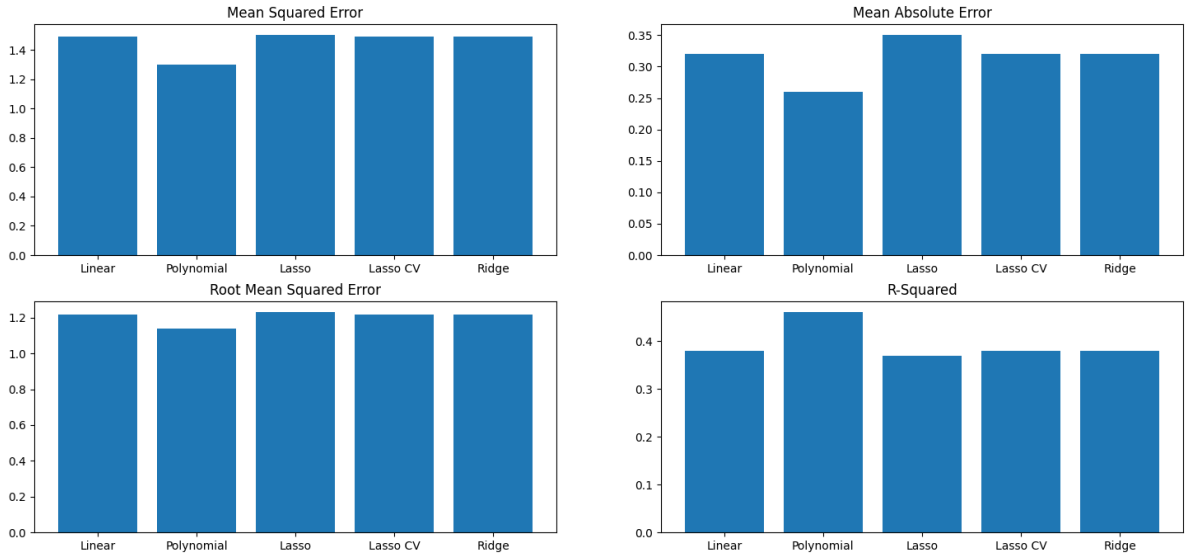


Figura 26. Comparação modelos de regressão

## DASHBOARD – POWER BI

A ferramenta utilizada para análise da base de dados foi aplicação Microsoft Power BI Desktop. Este é um software que permite o tratamento, modelagem e a visualização as bases de dados, em paneis, permitindo criar relatórios de dados de forma dinâmica e interativa, e assim, fornecer insights importantes para a tomada de decisões.

Para analisar os resultados da campanha, foi elaborado um relatório, com dois paneis. No primeiro, é possível fazer a análise pelas características dos clientes do banco, como o nível de educação e a área de formação. Ainda no primeiro painel é possível efetuar a análise por mês, ou a relação entre a idade do cliente e o saldo à data do contacto telefónico. No segundo painel, é possível fazer análise à adesão, ou não da campanha.

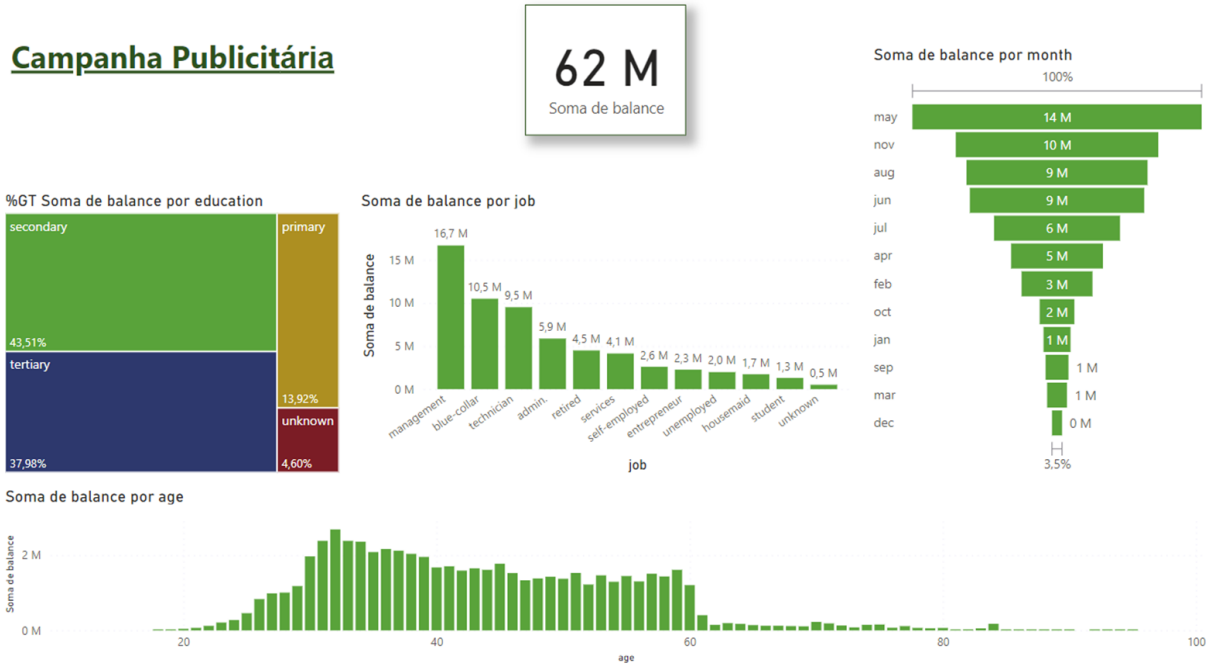


Figura 27. *Dashboard* – Painel 1

Através do primeiro painel, e fazendo uma breve análise aos dados estáticos, conseguimos concluir que mais de 80% dos clientes contactados têm uma educação secundária e terciária, e que os clientes mais contactados têm a profissão de gestão. Podemos ainda concluir, através do

segundo painel que aproximadamente 85% dos inquiridos optou por não subscrever a oferta. Dos clientes que subscreveram a oferta, mais de 90% não possuía um empréstimo à data.

Foi ainda elaborada uma análise, utilizando um gráfico de dispersão entre as variáveis *'age'* e *'balance'*, não tendo sido possível a obtenção de grupos, ou clusters.

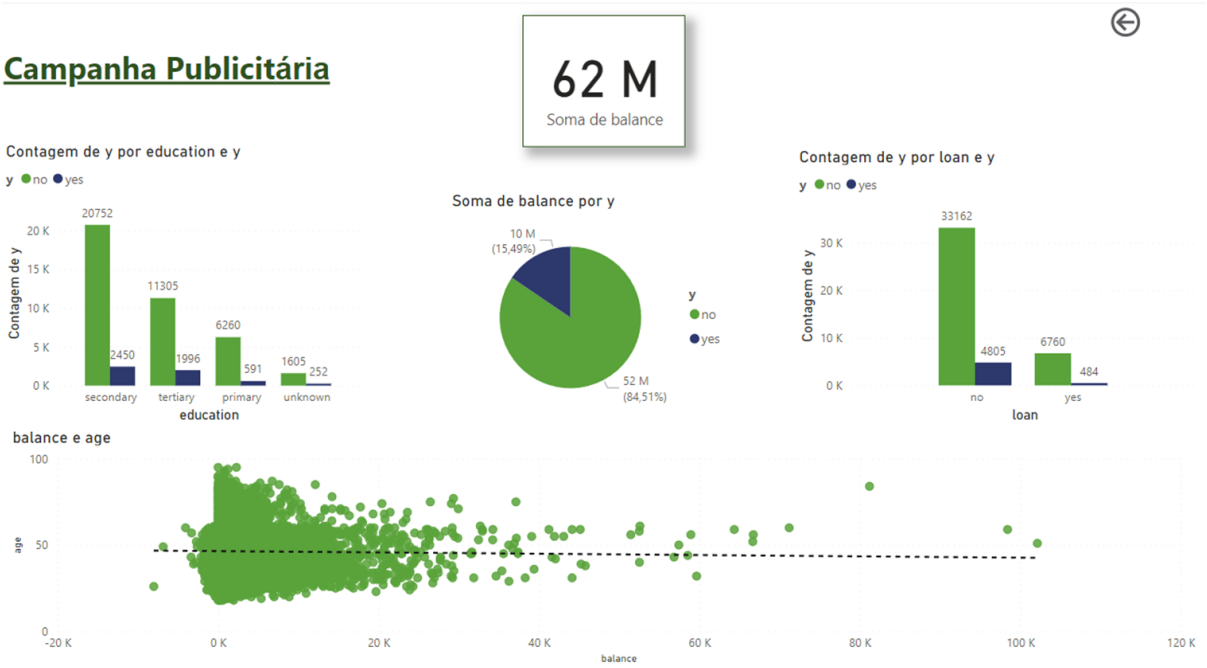


Figura 28. *Dashboard* – Painel 2

## CONCLUSÕES

Conforme previamente introduzido, este projeto teve como objeto a análise de um *dataset* previamente definido, este contendo dados acerca do resultado de uma campanha de adesão a novo produto/produto de crédito, assim como dados relativamente aos atributos da carteira de clientes que foi considerada.

Esta análise foi realizada com base nos diversos modelos introduzidos ao longo do decurso da cadeira de *Machine Learning* do curso de pós-graduação em Data Science, tendo-se intentado a introdução de cada um dos modelos presentes no módulo, divididos nas categorias: associação, *clustering*, classificação e regressão, assim como breve análise com uso da ferramenta *Microsoft Power BI*.

Relativamente à análise por modelos de associação, foi considerado o modelo apriori, onde se pode verificar que a associação mais forte será onde uma pessoa tenha respondido "*aug*" e "no", terá "*cellula*r", com uma confiança de 0,98. Já o lift de 1,5 indica-nos que a resposta de "*cellular*" tem 1,5 vezes mais hipótese de acontecer do que se as respostas antecedentes forem outras qualquer.

Considerando modelos de *clustering*, verificou-se que o *dataset* escolhido apresenta uma vasta dispersão de atributos, sendo que, na aplicação dos 3 (três) métodos considerados, não tendo sido possível a obtenção de grupos, ou *clusters*, mutuamente distintos.

Os modelos de classificação apresentam um comportamento uniforme entre os métodos utilizados, tendo-se obtido, globalmente, graus de precisão entre 76% e 90%. Destes, destacam-se: *ADABOOST*, Árvore de Decisão e NB (*Gaussian*); apresentando elevado grau de precisão face aos restantes métodos e maior grau de verdadeiros positivos e verdadeiros negativos quando aplicada a matriz de confusão.

Por fim, nos modelos de regressão podemos verificar que o R-Squared varia entre 0.37 e 0.46, indicando que 37% a 46% da variabilidade da variável dependente pode ser explicada pela variável independente. O erro absoluto médio (MAE) indica-nos que os valores preditos divergem entre 0,26 e 0,35 dos valores reais. Já a validação cruzada dos modelos é de 0,21, ou seja, os modelos explicam 21% da variância da variável dependente nos diferentes subsets. Contudo, também temos um desvio-padrão relativamente alto de 0,18 o que pode sugerir que a performance dos modelos pode variar muito dependendo do subset de dados usado e também

pode ser um indicador de overfitting, ou seja, o modelo pode ter uma boa performance nos dados de treino, mas uma péssima performance em dados novos.

Relativamente a dificuldades encontradas durante o desenvolvimento do presente estudo, em primeiro lugar destaca-se o tamanho da base de dados, devido à necessidade de importação, através da biblioteca Pandas, e tratamento de um volume ascendendo a 47 mil entradas, o que, ao utilizar modelos de análise como, por exemplo, métodos de agrupamento e *clustering*, resulta num tempo de processamento elevado, sendo por vezes necessário executar o script durante várias horas para obtenção de resultados de forma a serem retiradas conclusões. Para o efeito, conforme acima mencionado, entendeu-se a retirada aleatória de uma amostra da base de dados, de maneira a reduzir o tempo de execução, tentando-se manter o máximo de integridade dos dados. Adicionalmente, o não conhecimento integral da iniciativa que compõe a base de dados escolhida, assim como a vasta dispersão e não uniformidade desta, tornaram-se um fator de dificuldade aquando da escolha dos atributos a considerar para a aplicação dos modelos.

Com base nos fatos observados, face à obtenção de um grau elevado de precisão obtido através da análise nesses modelos e à reduzida fidelidade visual e de conclusões nos restantes, é possível concluir que os modelos de classificação são aqueles que melhor explicam a base de dados considerada para o projeto.

## BIBLIOGRAFIA

TDH (2022). Interpretation of Evaluation Metrics For Regression Analysis (MAE, MSE, RMSE, MAPE, R-Squared, And Adjusted R-Squared). Obtido em 12 de abril de 2023, de https://traindatahub.com/?p=291

Scikit-learn - Machine Learning in Python. Disponível em https://scikit-learn.org/stable/. Acesso em abril de 2023.