Instituto Politécnico de Setúbal

Escola Superior de Tecnologia do Barreiro

Projeto de "Big Data"

Licenciatura em Bioinformática

Análises do *Dataset* "Viabilidade de Empréstimos"

janeiro de 2022

Grupo 5
André Antunes (201900177)
Nuno Melo (201700465)

Índice

1 Introdução **1**

2 Análise dos Dados **2**
2.1 Descrição 2
2.2 Limpeza Inicial dos Dados 3
2.3 Análise Exploratória 5

3 Algoritmos de Aprendizagem Automática **10**
3.1 Random Forest 14
3.2 Regressão Logística 16

4 Conclusão **18**

Referências **19**

1 Introdução

Este projecto consiste em analisar um conjunto de dados cedidos pela docente, relativos a empréstimos realizados por um banco, utilizando técnicas de *Big Data* e *Machine Learning*.

Um cliente incorre em incumprimento quando não paga os empréstimos no prazo estipulado, o que origina perdas significativas ao banco. Por este motivo, todos os anos os bancos têm perdas que impactam o crescimento económico do país. Para colmatar este problema, os bancos atribuem uma pontuação de crédito que indica a probabilidade do cliente conseguir pagar um empréstimo. Tenciona-se assim, com base num conjunto de características, prever o desfecho dos empréstimos [1].

Neste trabalho pretende-se efectuar transformações aos dados e a sua respetiva descrição, fazendo uma limpeza prévia e uma análise exploratória aos dados, e ainda a aplicação de dois algoritmos de *Machine Learning* utilizando o PySpark.

O PySpark é a colaboração entre o Apache Spark e o Python. Apache Spark é uma estrutura de computação em cluster de código aberto, construída em torno da velocidade e da facilidade de uso, enquanto que, o Python é uma linguagem de programação de alto nível de uso geral [2].

Figura 1: PySpark [3].

PySpark é uma ótima linguagem para realizar análises exploratórias de dados em escala e construir pipelines de *Machine Learning*, sendo uma alternativa à biblioteca Pandas, então utiliza-se o PySpark a fim de criar análises e *pipelines* mais escaláveis [4].

2 Análise dos Dados

2.1 Descrição

Como já foi dito anteriormente, irá utilizar-se um conjunto de dados relativos a empréstimos de um banco. Este conjunto de dados contém 33 atributos, dos quais 24 são valores numéricos (15 inteiros e 9 decimais) e 9 valores categóricos. E ainda, existem 27463 amostras, como mostra a figura 2.

```
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   ID                           27463 non-null  int32
 1   Loan_Amount                  27463 non-null  int32
 2   Funded_Amount                27463 non-null  int32
 3   Funded_Amount_Investor       27463 non-null  float64
 4   Term                         27463 non-null  int32
 5   Batch_Enrolled               27463 non-null  object
 6   Interest_Rate                27449 non-null  float64
 7   Grade                        27463 non-null  object
 8   Sub_Grade                    27463 non-null  object
 9   Home_Ownership               27463 non-null  object
 10  Employment__Duration         27463 non-null  float64
 11  Verification_Status          27463 non-null  object
 12  Payment_Plan                 27463 non-null  object
 13  Loan_Title                   27463 non-null  object
 14  Debit_to_Income              27463 non-null  float64
 15  Delinquency_-_two_years      27463 non-null  int32
 16  Inquires_-_six_months        27463 non-null  int32
 17  Open_Account                 27463 non-null  int32
 18  Public_Record                27463 non-null  int32
 19  Revolving_Balance            27463 non-null  int32
 20  Revolving_Utilities          27463 non-null  float64
 21  Total_Accounts               27463 non-null  int32
 22  Initial_List_Status          27463 non-null  object
 23  Total_Received_Interest      27463 non-null  float64
 24  Total_Received_Late_Fee      27463 non-null  float64
 25  Recoveries                   27463 non-null  float64
 26  Collection_Recovery_Fee      27463 non-null  float64
 27  Application_Type             27463 non-null  object
 28  Last_week_Pay                27463 non-null  int32
 29  Total_Collection_Amount      27463 non-null  int32
 30  Total_Current_Balance        27463 non-null  int32
 31  Total_Revolving_Credit_Limit 27463 non-null  int32
 32  Loan_Status                  27463 non-null  int32
dtypes: float64(9), int32(15), object(9)
```

Figura 2: Tipos de atributos.

2.2 Limpeza Inicial dos Dados

Antes da análise exploratória dos dados é necessário proceder à limpeza dos dados, removendo ou alterando dados incompletos, duplicados ou irrelevantes para a análise. Primeiro, começou-se por substituir os espaços que existiam nos nomes dos atributos por um *underscore*, apenas por uma questão de comodidade. Verifica-se que o atributo "Home_Ownership"e o atributo "Employment_Duration"estão trocados, devido aos valores de cada um deles procede-se à sua troca, figura 3.

Rename columns `Home_Ownership` and `Employment_Duration` that were swapped.

```
dataset = dataset.withColumnRenamed('Home_Ownership','Employment__Duration')\
                 .withColumnRenamed('Employment_Duration','Home_Ownership')
```

Figura 3: Renomear atributos "Home_Ownership"e "Employment_Duration".

Em seguida, converteu-se o *dataset* para a biblioteca Pandas, utilizando o comando .toPandas(), para utilizar funções que não estão integradas no PySpark. Começou-se por observar se existiam valores omissos, em que foram encontrados em apenas um dos atributos, figura 8. Decidiu-se retirar os valores omissos por existirem em pouca quantidade (14) no atributo "Interest_Rate", figura 4.

```
dataset_pd = dataset_pd.dropna()
print('Sum of NA values in Interest_Rate:',dataset_pd['Interest_Rate'].isnull().sum())
```

```
Sum of NA values in Interest_Rate: 0
```

Figura 4: Remoção dos valores omissos.

Na verificação efectuada aos dados, percebeu-se que os valores do atributo "Loan_Title" tinham muitas incongruências, valores idênticos com escrita díspar, 108 valores únicos neste atributo. Efectuou-se então a reorganização de valores similares para um valor geral obtendo no final 12 valores únicos no atributo "Loan_Title", como mostra na figura 5.

The attribute `Loan_Title` has now only 12 values names.

```
value_counts = dataset_pd['Loan_Title'].nunique()
print(value_counts)
```

```
12
```

Figura 5: Número total de valores únicos no atributo "Loan_Title".

Em seguida, observou-se que o atributo "Payment_Plan" tinha um valor único, figura 6. Removeu-se os atributos "Payment_Plan" e "ID" porque não teriam relevância para o algoritmo de *Machine Learning*, figura 7.

```
payment_plane_unique = dataset_pd['Payment_Plan'].unique()
print(f'Payment_Plan unique value: {payment_plane_unique}')
```

```
Payment_Plan unique value: ['n']
```

Figura 6: Valor único do atributo "Payment_Plan".

```
dataset_pd = dataset_pd.drop(['ID','Payment_Plan'], axis=1)
```

Figura 7: Remoção dos atributos "Payment_Plan" e "ID".

```
dataset_pd.isnull().sum()
```

ID	0
Loan_Amount	0
Funded_Amount	0
Funded_Amount_Investor	0
Term	0
Batch_Enrolled	0
Interest_Rate	14
Grade	0
Sub_Grade	0
Home_Ownership	0
Employment__Duration	0
Verification_Status	0
Payment_Plan	0
Loan_Title	0
Debit_to_Income	0
Delinquency_-_two_years	0
Inquires_-_six_months	0
Open_Account	0
Public_Record	0
Revolving_Balance	0
Revolving_Utilities	0
Total_Accounts	0
Initial_List_Status	0
Total_Received_Interest	0
Total_Received_Late_Fee	0
Recoveries	0
Collection_Recovery_Fee	0
Application_Type	0
Last_week_Pay	0
Total_Collection_Amount	0
Total_Current_Balance	0
Total_Revolving_Credit_Limit	0
Loan_Status	0

dtype: int64

Figura 8: Presença de valores omissos.

2.3 Análise Exploratória

Após a limpeza de dados, onde retirou-se valores omissos e corrigiu-se dados incoerentes, pôde-se prosseguir para a análise exploratória de dados. Esta fase ajudou a identificar padrões entre os atributos, detectar *outliers* ou ainda, ajudar na identificação de erros que ainda não foram observados. Começou-se por observar a distribuição no atributo "Loan_Status", o atributo *target* no conjunto de dados, demonstrado na figura 9. Observou-se que o valor 0 apresenta grande parte da distribuição, contendo 24926 valores (aproximadamente 91%) , enquanto o valor 1 apresenta 2523 valores (apenas aproximadamente 9%), no conjunto de dados. Assumiu-se que o valor 1 representa os clientes infractores, que não cumpriram o pagamento do empréstimo e o valor 0 representa os clientes que cumpriram o pagamento do empréstimo ao banco. Logo, a grande maioria dos clientes cumprirá o pagamento do empréstimo.

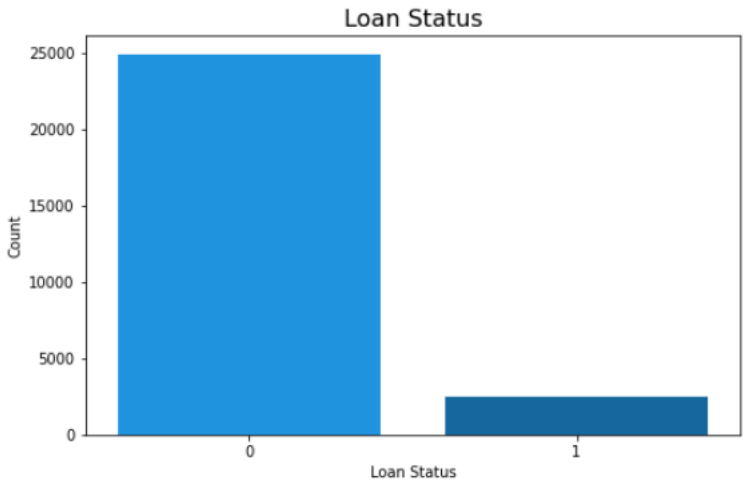

Figura 9: Distribuição do "Loan_Status".

De seguida efetuou-se uma combinação entre os atributos "Loan_Status"e o "Grade". Achamos que o *Grade* tem como o significado de uma nota atribuída pelo banco, a um certo empréstimo. Com base na figura 10, verificou-se que existe uma grande distribuição sobretudo nas notas B e C, enquanto há poucos dados que correspondem às notas F e G. Um outro atributo que combinou-se com o atributo "Grade"foi o "Home_Ownership", do tipo categórico. Tem como valores: *Mortgage*, *Rent* e *Own*. Na figura 11 observou-se que os clientes, na sua maioria têm a casa hipotecada. Também se consegue perceber pela análise da figura 10, que os clientes receberam notas altas, A, B e C, e que a maior parte deles não cumprirá com o pagamento do empréstimo.

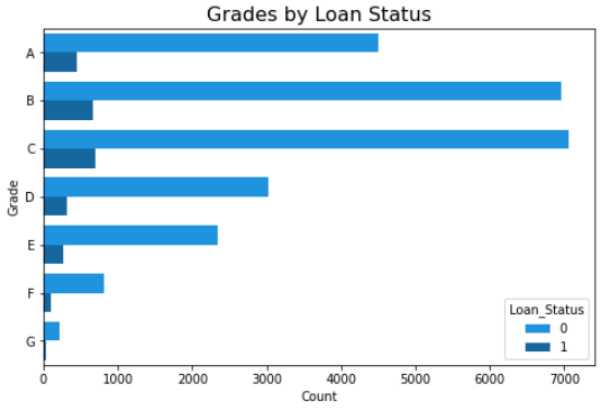

Figura 10: Distribuição do "Grades"por "Loan_Status".

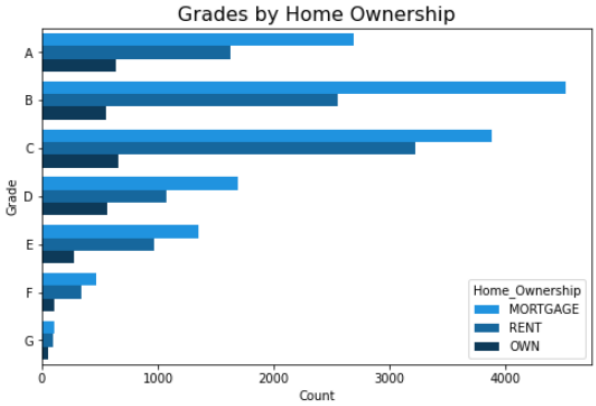

Figura 11: Distribuição do "Grades"por "Home_Ownership".

De seguida, analisou-se a variável "Verification_Status"que tinha três valores, *Verified*, *Not Verified* e *Source Verified*, figura 12, a que se referem se o cliente já foi verificado pelo banco em relação ao empréstimo ou não. Na nossa perspectiva, o *Verified* e o e *Source Verified* correspondiam ao mesmo, e decidiu-se agregar ambas no valor *Verified*, 13. Após a alteração, 76% dos clientes já foram verificados, enquanto que 24% não foram verificados.

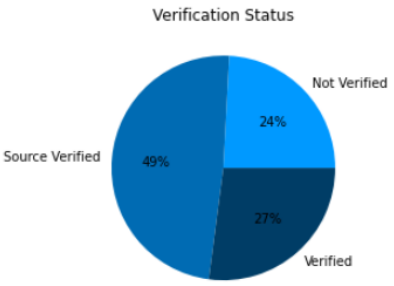

Figura 12: "Verification_Status" antes da alteração.

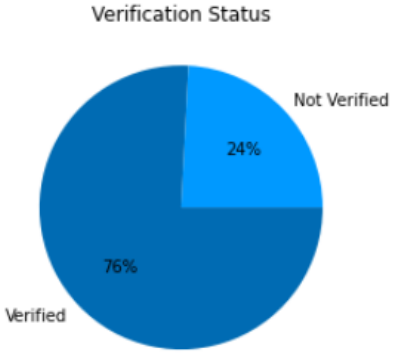

Figura 13: "Verification_Status" depois da alteração.

Depois, observou-se a variação do "Loan_Amount" por "Loan_Title", na figura 14, onde evidenciou-se imediatamente a existência de um *outlier* no *Moving and relocation*. Por isso, visualizou-se num *boxplot* a distribuição do "Loan_Amount" apenas, figura 15, não existindo nenhum *outlier* concluiu-se que o valor elevado no *Moving and relocation* não era um verdadeiro *outlier*, por isso não foi removido. Nos restantes valores do "Loan_Title" não existe uma grande assimetria dos valores presentes em cada um, excepto no valor *Vacation* e no *Personal Loan*.

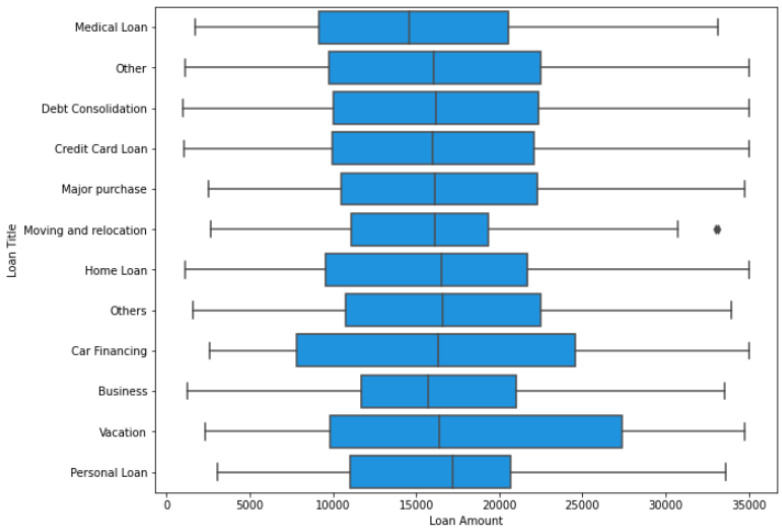

Figura 14: *Boxplot* do "Loan_Amount" por "Loan_Title".

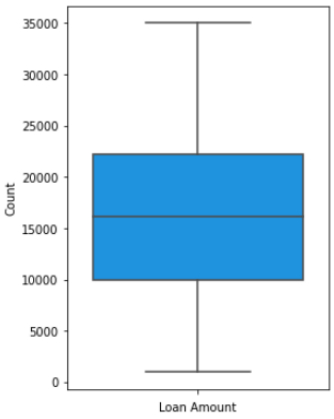

Figura 15: *Boxplot* do "Loan_Amount".

De seguida, observou-se a variável "Application_Type"por o "Loan_Amount", como é demonstrado na figura 16, verificou-se que a variável é constituída por: *Joint* e o *Individual*.

No *boxplot*, notámos que existe uma assimetria no *Joint*, enquanto no *Individual* verificou-se uma simetria. Sendo que o *INDIVIDUAL* corresponde 99,7% da distribuição, enquanto o *JOINT* apenas corresponde a 0,03% (o que corresponde a 62 linhas).

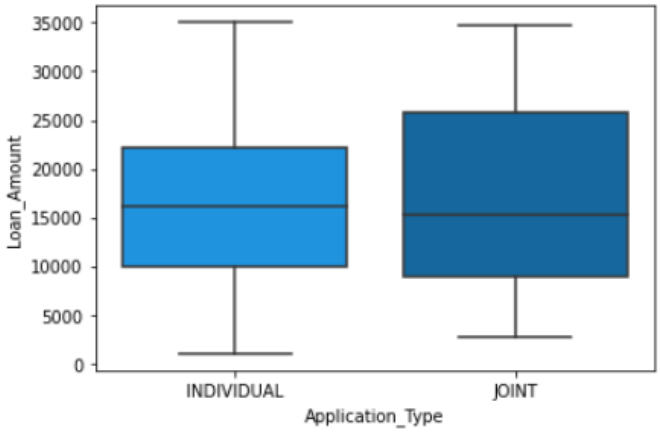

Figura 16: *Boxplot* do "Application_Type" por "Loan_Amount".

E por final, analisámos um *Heatmap*, apresentado na figura 17, para verificar se existe correlação entre todas as características numéricas do conjunto de dados. Podemos concluir, com base no *Heatmap*, que não existe qualquer semelhança significativa entre os atributos numéricos, o que é perfeito para optimizar os algoritmos de *machine learning*.

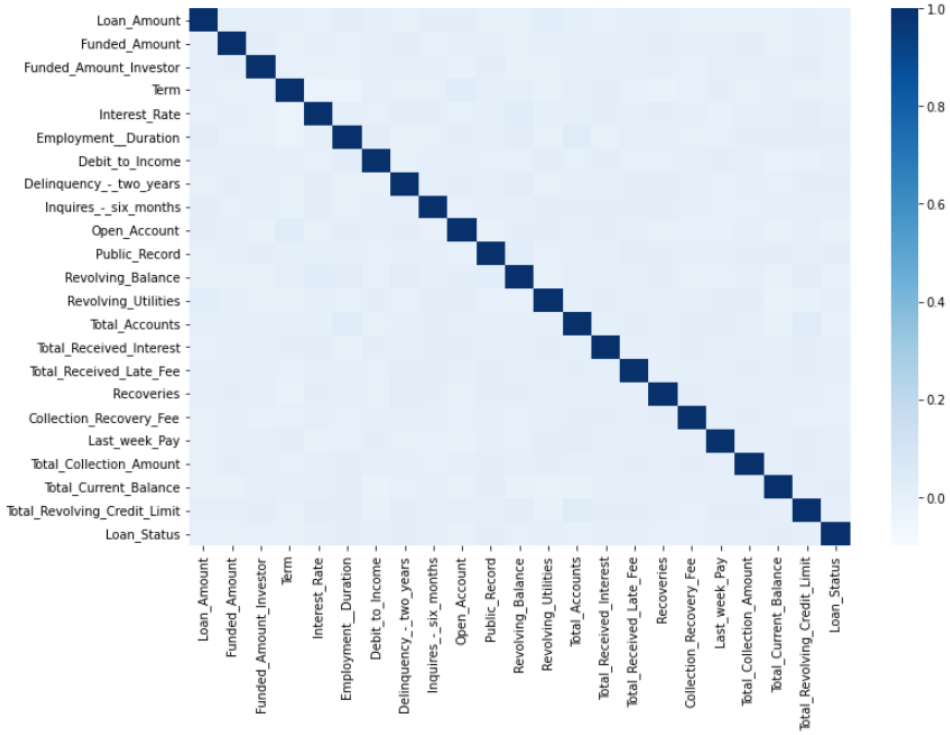

Figura 17: Correlação dos atributos.

3 Algoritmos de Aprendizagem Automática

Nesta etapa, pretendeu-se implementar alguns algoritmos de aprendizagem automática para avaliar o desempenho da classificação do atributo *target* "Loan_Status", em que o valor 1 representa os clientes infractores, que não cumpriram o pagamento do empréstimo e o valor 0 representa os clientes que cumpriram o pagamento do empréstimo ao banco.

Antes da implementação dos algoritmos, utilizou-se o método *LabelEncoder* que codifica valores dos atributos categóricos para valores numéricos, ação necessária para a aplicação dos modelos de aprendizagem automática. Em seguida, já com o *dataset* final, criou-se o novo *dataframe* no PySpark, com os valores de todos os atributos numéricos, figura 18.

As métricas utilizadas para avaliar os modelos foram a ***accuracy*** - que é dada pelo quociente entre o número de previsões corretas e o número total de elementos previstos, ou seja, o valor da exatidão da matriz de confusão, ***recall*** - representa a proporção de verdadeiros positivos e a ***precision*** - é dada pela capacidade do classificador gerar verdadeiros positivos.

Em seguida, utilizou-se o método *VectorAssembler*, como mostra a figura 19, que permite combinar vários atributos numa única coluna de vetor, *features*, que vai ser utilizado nos modelos de aprendizagem automática.

Depois, standardizou-se as *features* utilizando o *StandardScaler*, figura 20.

```
1 dataset.printSchema()
```

```
root
 |-- Loan_Amount: long (nullable = true)
 |-- Funded_Amount: long (nullable = true)
 |-- Funded_Amount_Investor: double (nullable = true)
 |-- Term: long (nullable = true)
 |-- Batch_Enrolled: long (nullable = true)
 |-- Interest_Rate: double (nullable = true)
 |-- Grade: long (nullable = true)
 |-- Sub_Grade: long (nullable = true)
 |-- Home_Ownership: long (nullable = true)
 |-- Employment__Duration: double (nullable = true)
 |-- Verification_Status: long (nullable = true)
 |-- Loan_Title: long (nullable = true)
 |-- Debit_to_Income: double (nullable = true)
 |-- Delinquency_-_two_years: long (nullable = true)
 |-- Inquires_-_six_months: long (nullable = true)
 |-- Open_Account: long (nullable = true)
 |-- Public_Record: long (nullable = true)
 |-- Revolving_Balance: long (nullable = true)
 |-- Revolving_Utilities: double (nullable = true)
 |-- Total_Accounts: long (nullable = true)
 |-- Initial_List_Status: long (nullable = true)
 |-- Total_Received_Interest: double (nullable = true)
 |-- Total_Received_Late_Fee: double (nullable = true)
 |-- Recoveries: double (nullable = true)
 |-- Collection_Recovery_Fee: double (nullable = true)
 |-- Application_Type: long (nullable = true)
 |-- Last_week_Pay: long (nullable = true)
 |-- Total_Collection_Amount: long (nullable = true)
 |-- Total_Current_Balance: long (nullable = true)
 |-- Total_Revolving_Credit_Limit: long (nullable = true)
 |-- Loan_Status: long (nullable = true)
```

Figura 18: *Schema* do *dataset* no PySpark.

Ainda antes de aplicar o algoritmo, procedeu-se à aplicação de uma técnica de *Undersampling*, que consiste em balancear as classes do atributo *target* "Loan_Status", onde existe uma grande assimetria na distribuição dos valores, figura 21.

Depois de aplicada a técnica, perdeu-se uma grande quantidade de dados, mas como o *dataset* é grande ainda manteve-se uma larga quantidade de dados, mas desta vez com as classes balanceadas, figura 22. Desta forma o algoritmo deverá prever com maior precisão as duas classes.

Combine *features* in a vector.

```
vector_assembler = VectorAssembler(inputCols=['Loan_Amount','Funded_Amount','Funded_Amount_Investor',
                                              'Term','Batch_Enrolled','Interest_Rate','Grade','Sub_Grade',
                                              'Home_Ownership','Employment__Duration','Verification_Status',
                                              'Loan_Title','Debit_to_Income','Delinquency_-_two_years',
                                              'Inquires_-_six_months','Open_Account','Public_Record',
                                              'Revolving_Balance','Revolving_Utilities','Total_Accounts',
                                              'Initial_List_Status','Total_Received_Interest',
                                              'Total_Received_Late_Fee','Recoveries','Collection_Recovery_Fee',
                                              'Application_Type','Last_week_Pay','Total_Collection_Amount',
                                              'Total_Current_Balance','Total_Revolving_Credit_Limit']
                                             ,outputCol="features")
output = vector_assembler.transform(dataset)
output.show(5)
```

Figura 19: *Vector Asembler* das *features* do *dataset*.

Standardizes the data.

```
from pyspark.ml.feature import StandardScaler

scaler = StandardScaler(inputCol="features", outputCol="scaledFeatures", withStd=True, withMean=False)

scalerModel = scaler.fit(output).transform(output)

scalerModel.select("features","scaledFeatures").show(5)
```

```
+--------------------+--------------------+
|            features|      scaledFeatures|
+--------------------+--------------------+
|[12368.0,23353.0,...|[1.47212500680719...|
|[12622.0,26273.0,...|[1.50235784572448...|
|[4728.0,9827.0,11...|[0.56275930079110...|
|[27493.0,16144.0,...|[3.27240724548433...|
|[21095.0,13706.0,...|[2.51087298015829...|
+--------------------+--------------------+
only showing top 5 rows
```

Figura 20: *Standard Scaler* das *features* do *dataset*.

```
+-----------+-----+
|Loan_Status|count|
+-----------+-----+
|          0|24926|
|          1| 2523|
+-----------+-----+
```

Figura 21: Distribuição antes do *Undersampling*.

```
+-----------+-----+
|Loan_Status|count|
+-----------+-----+
|          0| 2835|
|          1| 2523|
+-----------+-----+
```

Figura 22: Distribuição depois do *Undersampling*.

Em seguida, aplicou-se o *train test split* com 70% para o treino e 30% para o teste, figura 23.

```
1 train, test = final_data.randomSplit([0.70, 0.30])
2 print("training dataset:", str(train.count()))
3 print("test dataset:", str(test.count()))
```

```
training dataset: 3755
test dataset: 1603
```

Figura 23: Divisão do *dataset* em treino e teste.

3.1 Random Forest

O nosso primeiro algoritmo será o *Random Forest*, na figura 24, é demonstrado o *script* do modelo, e o resultado da *accuracy*, *recall* e *precision*, na figura 25, do algoritmo.

```
from pyspark .ml.classification import RandomForestClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator


random_forest = RandomForestClassifier(featuresCol='features', labelCol='labelIndexed')
rfModel = random_forest.fit(train)
predictions = rfModel.transform(test)
predictions.select("prediction", 'labelIndexed').show(5)
```

```
+----------+------------+
|prediction|labelIndexed|
+----------+------------+
|       1.0|         0.0|
|       1.0|         0.0|
|       0.0|         0.0|
|       1.0|         0.0|
|       1.0|         0.0|
+----------+------------+
only showing top 5 rows
```

Figura 24: *Script* do modelo *Random Forest*.

```
print(classification_report(test.select('labelIndexed').toPandas(),predictions.select('prediction').toPandas()))
```

```
              precision    recall  f1-score   support

         0.0       0.53      0.75      0.62       805
         1.0       0.51      0.28      0.36       748

    accuracy                           0.52      1553
   macro avg       0.52      0.51      0.49      1553
weighted avg       0.52      0.52      0.49      1553
```

Figura 25: *Classification Report* do *Random Forest*.

Podemos concluir que o algoritmo, *Random Forest*, tem como accuracy 52%, o que significa que o modelo encontra-se com um desempenho muito fraco, considerando que a percentagem de erro no modelo é aproximadamente 48%.

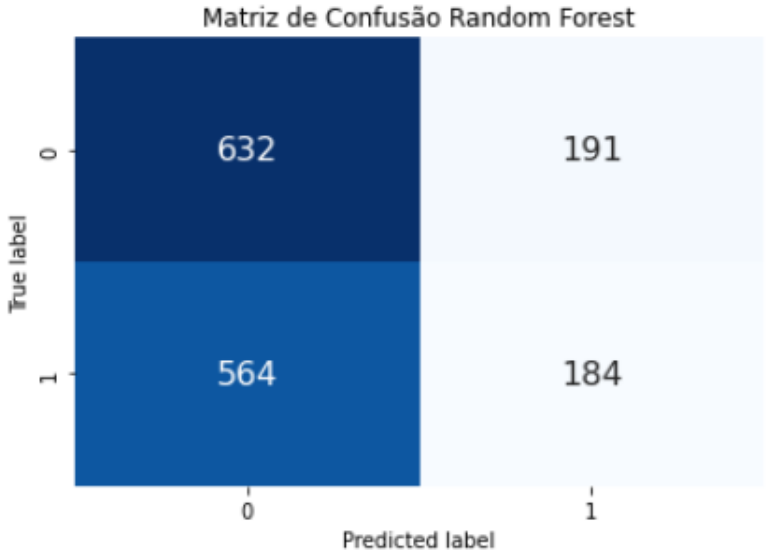

Figura 26: *Confusion Matrix* do *Random Forest*.

Com base na matriz de confusão, representada na figura 26, concluímos com as seguintes alíneas que:

- *True Positives* - 184
- *True Negatives* - 632
- *False Positives* - 191
- *False Negatives* - 564

Observamos que os *True Negatives* apresentam 40% da distribuição dos dados, sendo a maior percentagem, enquanto os *True Positives* apresentam apenas 11%, sendo a menor percentagem na distribuição.

3.2 Regressão Logística

Para o segundo algoritmo utilizou-se a Regressão Logística, a divisão dos dados é igual à anterior, 70% para o treino e 30% para o teste. Este modelo, figura 27 recebeu como parâmetro o *scaledFeatures* que são todas as *features* standardizadas e o atributo *target* "Loan_Status".

Apresentou uma *accuracy* para o conjunto de treino de aproximadamente 56% e para o conjunto de teste de 53%, figura 28. É um desempenho relativamente fraco, mesmo depois de termos balanceado as classes.

A figura 29 mostra o resultado da *accuracy*, *recall* e da *precision* obtidas para o algoritmo. Removeu-se ainda algumas *features* para tentar melhorar a *accuracy*, mas manteve-se semelhante.

```
lr = LogisticRegression(featuresCol = "scaledFeatures",labelCol="Loan_Status", maxIter=10)
lrModel = lr.fit(train)
predict_train=lrModel.transform(train)
predict_test=lrModel.transform(test)
```

Figura 27: *Script* do modelo da Regressão Logística.

```
evaluator=BinaryClassificationEvaluator(rawPredictionCol="rawPrediction",labelCol="Loan_Status")
predict_test.select("Loan_Status","rawPrediction").show(5)

print("Train score {}".format(evaluator.evaluate(predict_train)))
print("Test score {}".format(evaluator.evaluate(predict_test)))

+-----------+--------------------+
|Loan_Status|       rawPrediction|
+-----------+--------------------+
|          0|[-0.0093076351155...|
|          0|[-0.2205524899260...|
|          0|[-0.1144591087626...|
|          0|[0.12201258387292...|
|          0|[-0.1440299298663...|
+-----------+--------------------+
only showing top 5 rows

Train score 0.5596953450699027
Test score 0.5324331702054884
```

Figura 28: *Accuracy* para treino e teste na Regressão Logística.

```
print(classification_report(test.select('Loan_Status').toPandas(),predict_test.select('prediction').toPandas()))
```

```
              precision    recall  f1-score   support

           0       0.53      0.75      0.62       844
           1       0.51      0.28      0.36       770

    accuracy                           0.53      1614
   macro avg       0.52      0.52      0.49      1614
weighted avg       0.52      0.53      0.50      1614
```

Figura 29: *Classification Report* da Regressão Logística.

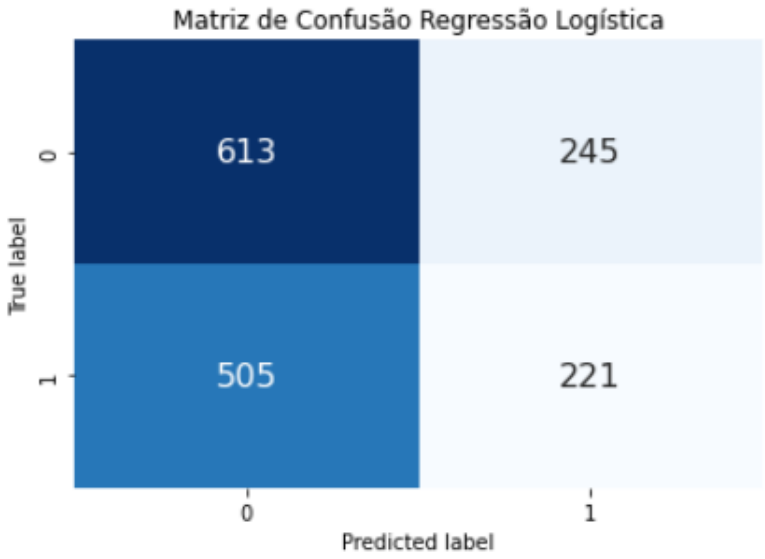

Figura 30: *Confusion Matrix* da Regressão Logística.

Com base na matriz de confusão, representada na figura 30, concluímos com as seguintes alíneas que:

- *True Positives* - 221
- *True Negatives* - 613
- *False Positives* - 245
- *False Negatives* - 505

Observamos que os *True Negatives* apresentam, aproximadamente, 39% na distribuição dos dados, sendo a maior percentagem, enquanto os *True Positives* apresentam apenas 14%, sendo a menor percentagem na distribuição.

4 Conclusão

Os bancos têm perdas significativas quando os clientes não pagam os empréstimos no prazo estipulado. É possível prever, com base num conjunto de características, se o cliente pagará os empréstimos no prazo ou não. Para isto, efectuou-se uma análise exploratória para encontrar padrões entre os atributos e erros que poderiam existir nos dados, e de seguida, aplicou-se algoritmos de aprendizagem automática para avaliar o desempenho da classificação do atributo *target* "Loan_Status".

No primeiro algoritmo, *Random Forest* verificou-se que o algoritmo iria classificar incorrectamente devido ao facto das classes do atributo *target* "Loan_Status" não estarem bem distribuídas, aplicou-se então uma técnica de *Undersampling*, que consiste em reduzir a classe maior para que as classes fiquem balanceadas. Feitas estas alterações, aplicou-se o *train test split* e classificou-se utilizando o *Random Forest*. Obteve-se um desempenho fraco para este modelo, com *accuracy* de 52%, *precision* de 53% e 51% para o valor 0 e valor 1, respetivamente, e com *recall* de 75% para o valor 0 e apenas de 28% para o valor 1, o que indica que este algoritmo para este valor apresentou poucos verdadeiros positivos.

No segundo algoritmo, *Regressão Logística* obteve-se uma *accuracy* de aproximadamente 56% para o conjunto de treino e 53% para o conjunto de teste, ligeiramente superior ao *Random Forest* contudo o desempenho continua fraco.A *precision* de 53% e 51% para o valor 0 e valor 1, igual ao *Random Forest*, e com *recall* de 75% para o valor 0 e 28% para o valor 1.

Podemos concluir que os objetivos deste projeto foram cumpridos, apesar de não termos obtido bons resultados na classificação das classes da *target*. Este projeto foi importante para aprender a utilizar o PySpark, uma ferramenta importante quando trabalhamos com *Big Data*.

Referências

[1] “Key Things To Follow To Avoid Defaulting On Your Loans – Forbes Advisor INDIA.” [Online]. Available: https://www.forbes.com/advisor/in/loans/key-things-to-follow-to-avoid-defaulting-on-your-loans/

[2] “PySpark Programming — What is PySpark? — Introduction To PySpark — Edureka.” [Online]. Available: https://www.edureka.co/blog/pyspark-programming/

[3] “What is PySpark? - Databricks.” [Online]. Available: https://databricks.com/glossary/pyspark

[4] “A Brief Introduction to PySpark. — by Ben Weber — Towards Data Science.” [Online]. Available: https://towardsdatascience.com/a-brief-introduction-to-pyspark-ff4284701873