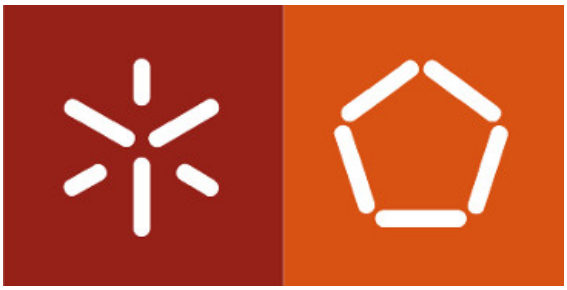

Universidade do Minho
Mestrado Integrado em Engenharia Informática
Ciências de Dados

Bank Marketing

Aprendizagem Automática II

Carolina Cunha, A80142
Bruno Veloso, A78352

6 de junho de 2021

Conteúdo

Capítulo 1

Introdução

As campanhas de vendas constituem uma estratégia típica para aumentar o volume de negócios. As empresas usam o marketing direto para entrar em contato com determinados clientes, de modo a atingir objetivos de vendas estipulados. A centralização das interações remotas com o cliente facilita a gestão operacional das campanhas. Essas interações dão-se por diversos meios, sendo o telefone (fixo ou móvel) um dos mais utilizados (telemarketing) [1].

1.1 Descrição do problema

O conjunto de dados utilizados pertence ao *dataset* **Bank Marketing**. Estes dados são referentes a campanhas de *marketing* direto de uma instituição bancária portuguesa, efetuadas via chamadas telefónicas, realizadas entre maio de 2008 e novembro de 2010. Ocasionalmente, foi necessário contactar o cliente por mais de uma vez, de modo a determinar o seu interesse no produto [2].

O objetivo deste estudo é prever corretamente se o cliente pretende assinar, ou se assinou, um depósito a prazo, assumindo o valor da variável de interesse Y. Deste modo, serão aplicados métodos de *Machine Learning*, bem como de *Deep Learning*, incluindo a otimização de hiperparâmetros, numa tentativa de melhorar os resultados obtidos.

1.2 Descrição do conjunto de dados

O conjunto de dados real escolhido é constituído por 41188 exemplos (linhas) e 21 atributos (colunas). De seguida serão enumeradas, detalhadamente, os atributos relativos ao cliente, ao último contacto da campanha atual, aos dados de contextos sociais e económicos e, ainda, de outros contextos. Por fim, será apresentada a variável de decisão.

1.2.1 Atributos

Cliente

- *age* – Idade do cliente (variável numérica);
- *job* – Tipo de emprego (variável categórica: *admin., blue-collar, entrepreneur, housemaid, management, retired, self-employed, services, student, technician, unemployed, unknown*);
- *marital* – Estado conjugal (variável categórica: *divorced, married, single, unknown*);
- *education* – Nível de Educação (variável categórica: *basic.4y, basic.6y, basic.9y, high.school, illiterate, professional.course, university.degree, unknown*);
- *default* – Crédito em Incumprimento (variável categórica: *yes, no, unknown*);
- *housing* – Empréstimo de habitação (variável categórica: *yes, no, unknown)*);

[1] Sérgio Moro, et al. «*A data-driven approach to predict the success of bank telemar-keting*» (2014)
[2] https://www.kaggle.com/henriqueyamahata/bank-marketing

- *loan* - Empréstimo pessoal (variável categórica: *yes, no, unknown)*);

Último contacto com a campanha atual

- *contact* – Tipo de comunicação (variável categórica: *cellular, telephone*);
- *month* – Último mês onde se realizou contacto (variável categórica: *mar, apr, may, jun, jul, aug, sep, oct, nov, dec*);
- *day_ of_ week* – Último dia da semana em que se realizou contacto (variável categórica: *mon, tue, wed, thu, fri*);
- *duration* – Duração do último contacto, em segundos (variável numérica);

Atributos de contextos sociais e económicos

- *emp.var.rate* – Taxa de variação de emprego, indicador trimestral (variável numérica);
- *cons.price.idx* – Índice de preço no consumidor, indicador mensal (variável numérica);
- *cons.conf.idx* – Índice de confiança no consumidor, indicador mensal (variável numérica);
- *euribor3m* – Taxa da euribor a três meses, indicador diário (variável numérica);
- *nr.employed* – Número de empregados, indicador trimestral (variável numérica);

Outros atributos

- *campaing* – Número de contactos realizados com um determinado cliente durante a campanha atual (variável numérica);
- *pdays* – Número de dias que passaram desde o último contacto com a campanha anterior (variável numérica, em que 999 indica que o cliente não foi contactado anteriormente);
- *previous* – Número de vezes que um mesmo cliente foi contactado antes da atual campanha (variável numérica);
- *poutcome* – Resultado da campanha de *marketing* anterior (variável categórica: *failure, nonexistent, success*);

1.2.2 Variável de interesse

Dada pela variável y, variável binária que indica se um cliente assinou o depósito a prazo.

1.3 Questões de interesse

1. Que atributos que mais favorecem a assinatura de um contrato?
2. Quais os modelos preditivos que conseguem ter uma melhor percentagem de acerto, no que diz respeito a saber se assinou, ou não, contrato?
3. Quais as classes de maior dificuldade de previsão sem balanceamento?

Capítulo 2

Preparação dos dados

2.1 Análise exploratória dos dados

De forma a estudar do conjunto de dados escolhido é necessário realizar uma análise exploratória dos dados, de modo a compreender o que significam, bem como as relações que apresentam entre si.

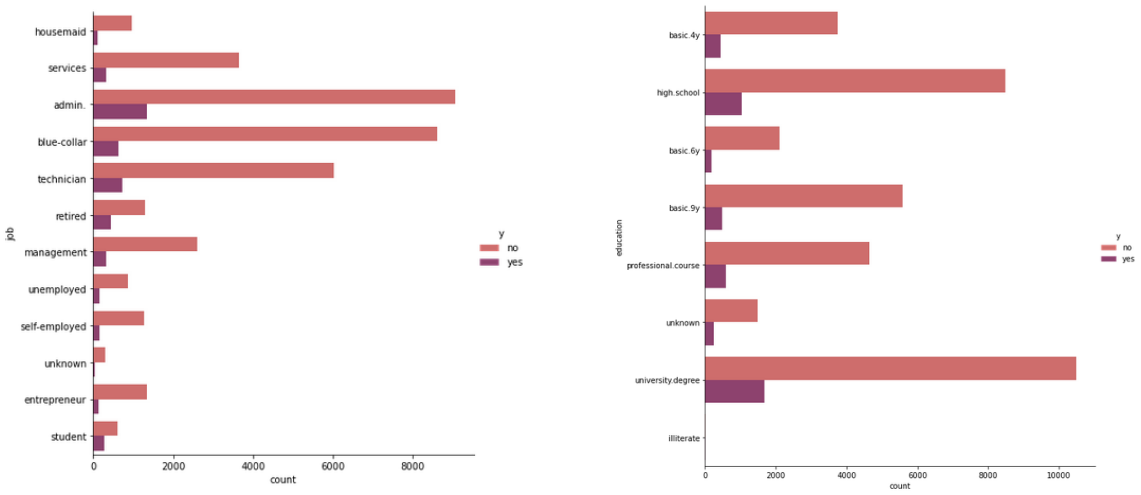

Figura 2.1: Correlação entre a variável de interesse e as variáveis *job* e *education*

Nos gráficos apresentados, observa-se a correlação entre a variável de interesse e os atributos emprego e nível de educação. Nestes, verifica-se uma grande discrepância na percentagem de clientes que assinam um depósito a prazo, independentemente do seu estatuto profissional ou educacional.

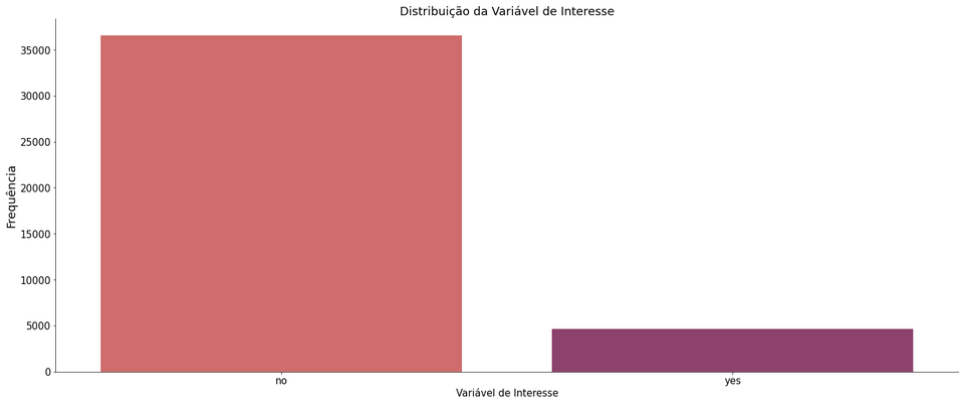

Figura 2.2: Distribuição da variável de interesse

A análise da distribuição da variável de interesse permite verificar uma disparidade significativa entre as classes 'sim' e 'não' dos dados recolhidos.

2.2 Pré-processamento dos dados

De modo a adequar a informação presente do *dataset* aos modelos de aprendizagem utilizados, foi realizado um pré-processamento dos dados. Neste, todos os atributos foram convertidos para valores numéricos, uma vez que os modelos de aprendizagem implementados trabalham sobre este tipo de valores. Foi dada especial atenção às variáveis *age*, cujos valores foram agrupados por faixa etária, originando quatro sub-grupos; e *duration*, agrupada e distribuída por cinco intervalos (Figura 2.3).

Figura 2.3: Divisão das variáveis *age* e *duration* em intervalos

Em conjuntos de dados do mundo real é frequente a existência de dados com valores inconsistentes ou nulos e de dados redundantes. Por este motivo, foram aplicadas três abordagens na concretização do pré-processamento dos dados. Destas abordagens surgiram os *datasets* que se seguem.

No primeiro *dataset*, foi utilizada a totalidade dos dados, sobre os quais foi aplicada uma normalização *StandardScaler*;

O segundo *dataset* teve todos os valores nulos (*unknown*) removidos, sendo igualmente aplicada uma normalização *StandardScaler*;

O terceiro *dataset* é idêntico ao anterior, no entanto, previamente à normalização dos dados (realizada com *StandardScaler*), foi aplicado *SMOTE*. Trata-se de uma técnica de *oversampling* em que as amostras sintéticas são geradas para a classe minoritária. Este algoritmo contribui para superar problemas de *overfitting* realizando uma sobreamostragem aleatória [1].

Para o quarto *dataset*, optou-se por, para além da remoção dos valores nulos, remover todas as colunas que apresentavam pouca correlação com a variável de interesse. Assim sendo, foram removidas as colunas *age, loan, month, housing, day_of_week* e, sobre os restantes dados, foi aplicada uma normalização com *StandarScaler*. Este *dataset* não é, no entanto, ideal, uma vez que o número de atributos é reduzido.

Por fim, no quinto *dataset* foi aplicado o método de substituição de valores nulos *KNNInputer* nos dados de treino. Através de algoritmos de k-Nearest Neighbours, pontos vizinhos são identificados através da distância entre estes, permitindo que os valores nulos sejam estimados através da média dos pontos vizinhos [2].

[1] https://www.analyticsvidhya.com/blog/2020/10/overcoming-class-imbalance-using-smote-techniques/
[2] https://www.analyticsvidhya.com/blog/2020/07/knnimputer-a-robust-way-to-impute-missing-values-using-scikit-learn/

Capítulo 3

Avaliação de Modelos

A avaliação dos modelos implementados é indispensável para determinar a sua capacidade de previsão de novos dados. A avaliação da qualidade de um modelo para uma dada tarefa é realizada calculando medidas de erro sobre um determinado conjunto de exemplos. Esta avaliação foi concretizada através de modelos de *Machine Learning*, com algoritmos de aprendizagem supervisionada e não supervisionada, e de *Deep Learning*.

3.1 Machine Learning

3.1.1 Aprendizagem Não Supervisionada

Nos paradigmas de aprendizagem automática não supervisionada, não é fornecida nenhuma indicação externa, sendo a aprendizagem realizada pela descoberta de regularidades nos dados de entrada.

No teor deste paradigma, foi utilizada a Análise de Componentes Principais (PCA), de forma a reduzir a dimensionalidade do *dataset*. Posteriormente, foi aplicado o algoritmo de *K-Means Clustering*, numa tentativa de agrupar as entradas por dois *clusters*.

y	no	yes
clusters		
0	8878	2906
1	17751	953

Figura 3.1: Resultados obtidos pelo algoritmo K-Means Clustering

Para além deste, foram aplicados algoritmos como o *Birch* e o *Gaussian Mixture*, presentes no repositório de trabalho.

3.1.2 Aprendizagem Supervisionada

Na aprendizagem automática supervisionada, é fornecida uma resposta correta para cada situação, e a aprendizagem é realizada a partir de exemplos compostos por um vetor de entradas e por um vetor de saídas desejadas. Em seguida, apresentam-se os resultados obtidos na aplicação dos diversos algoritmos de aprendizagem implementados.

Decision Tree

O primeiro algoritmo implementado recorreu às árvores de decisão. Este algoritmo utilizou o *GridSearchCV* para o otimização dos hiperparâmetros *criterion*, *max_depth* e *min_samples_leaf*. De forma a selecionar o melhor intervalo de valores para *max_depth*, foi construído um gráfico onde se observa a taxa de erro consoante a profundidade da árvore. O intervalo selecionado é aquele que apresenta menor taxa de erro. Os resultados do modelo aplicado aos cinco *datasets* apresentados encontram-se na tabela 3.2.

Datasets	Precision 0	Precision 1	Recall 0	Recall 1	f1_score 0	f1_score 1	Accuracy
D1	0.92	0.71	0.98	0.37	0.95	0.49	0.912
D2	0.93	0.61	0.95	0.53	0.94	0.57	0.898
D3	0.96	0.59	0.93	0.71	0.94	0.64	0.898
D4	0.91	0.63	0.98	0.24	0.95	0.35	0.900
D5	0.94	0.63	0.95	0.55	0.94	0.59	0.901

Figura 3.2: Resultados obtidos pelo algoritmo Decision Tree

Gradient Boosting

No que diz respeito ao algoritmo *Gradient Boosting*, foi também realizada a otimização de hiperparâmetros através do *GridSearchCV*. Os hiperparâmetros selecionados foram *loss*, *learning_rate*, *min_samples_split*, *min_samples_leaf*, *max_depth*, *max_features*, *criterion*, *subsample* e *n_estimators*. De igual forma ao sucedido no algoritmo anteriormente implementado, foi construído um gráfico da taxa de erro consoante a profundidade, de forma a selecionar um intervalo de valores para o parâmetro *max_depth*. Na tabela 3.3, são visíveis os resultados da aplicação do modelo aos cinco *datasets* anteriores.

Datasets	Precision 0	Precision 1	Recall 0	Recall 1	f1_score 0	f1_score 1	Accuracy
D1	0.94	0.66	0.97	0.50	0.95	0.57	0.915
D2	0.93	0.68	0.96	0.52	0.95	0.59	0.909
D3	0.95	0.61	0.94	0.68	0.94	0.64	0.902
D4	0.91	0.64	0.98	0.26	0.95	0.37	0.902
D5	0.94	0.65	0.95	0.57	0.95	0.61	0.906

Figura 3.3: Resultados obtidos pelo algoritmo Gradient Boosting

K-Nearest-Neighbors

A implementação do algoritmo *K-Nearest-Neighbors* foi idêntica aos anteriores. Neste, os hiperparâmetros a otimizar são *n_neighbors*, *weights* e *metric*. O intervalo de valores de K (parâmetro *n_neighbors*) mais apropriado foi obtido através da construção de um gráfico da taxa de erro consoante o valor de K, sendo escolhido aquele onde a taxa de erro é menor. Este algoritmo foi aplicado aos cinco *datasets* criados (tabela 3.4).

Datasets	Precision 0	Precision 1	Recall 0	Recall 1	f1_score 0	f1_score 1	Accuracy
D1	0.92	0.69	0.98	0.29	0.95	0.4	0.905
D2	0.90	0.71	0.98	0.26	0.94	0.38	0.893
D3	0.93	0.55	0.94	0.50	0.93	0.53	0.883
D4	0.91	0.64	0.98	0.22	0.95	0.33	0.900
D5	0.91	0.65	0.97	0.34	0.94	0.45	0.892

Figura 3.4: Resultados obtidos pelo algoritmo K-Nearest-Neighbors

Logistic Regression

Para este algoritmo, o *GridSearchCV* foi utilizado apenas para otimização do hiperparâmetro *C*. Os resultados deste algoritmo encontram-se na tabela que se segue.

Datasets	Precision 0	Precision 1	Recall 0	Recall 1	f1_score 0	f1_score 1	Accuracy
D1	0.92	0.68	0.98	0.37	0.95	0.48	0.910
D2	0.92	0.68	0.97	0.38	0.94	0.49	0.900
D3	0.94	0.59	0.94	0.56	0.94	0.58	0.893
D4	0.91	0.69	0.99	0.19	0.95	0.30	0.901
D5	0.92	0.67	0.97	0.42	0.94	0.52	0.899

Figura 3.5: Resultados obtidos pelo algoritmo Logistic Regression

Naive-Bayes

Optou-se, ainda, por implementar um algoritmo de *Naive-Bayes*, baseado no tratamento de probabilidades condicionais. Para a sua implementação, recorreu-se à variante *Gaussian Naive-Bayes*, tendo sido obtido os seguintes resultados.

Datasets	Precision 0	Precision 1	Recall 0	Recall 1	f1_score 0	f1_score 1	Accuracy
D1	0.95	0.38	0.87	0.60	0.91	0.46	0.843
D2	0.99	0.16	0.24	0.97	0.39	0.27	0.336
D3	0.98	0.15	0.20	0.98	0.33	0.27	0.300
D4	0.95	0.38	0.88	0.60	0.91	0.47	0.847
D5	0.99	0.16	0.24	0.99	0.39	0.28	0.336

Figura 3.6: Resultados obtidos pelo algoritmo Naive-Bayes

Random Forest

O algoritmo *Random Forest* é um método de *ensemble* de árvores de decisão. Este método ajusta diversos classificadores de árvores de decisão em diversas sub-amostras do *dataset*, usando a média para melhorar a previsão preditiva e controlar o *overfitting*. Na implementação deste algoritmo, foi utilizado o *GridSearchCV*, de forma a otimizar os parâmetros *bootstrap*, *max_ depth* e *n_ estimators*. Assim como previamente descrito, o intervalo de *max_ depth* é proveniente da análise do gráfico construído.

Datasets	Precision 0	Precision 1	Recall 0	Recall 1	f1_score 0	f1_score 1	Accuracy
D1	0.93	0.69	0.98	0.41	0.95	0.51	0.913
D2	0.92	0.68	0.97	0.45	0.95	0.54	0.904
D3	0.94	0.61	0.94	0.62	0.94	0.62	0.900
D4	0.91	0.66	0.99	0.20	0.95	0.31	0.900
D5	0.93	0.66	0.96	0.51	0.95	0.57	0.903

Figura 3.7: Resultados obtidos pelo algoritmo Random Forest

Support Vector Machines

No algoritmo de *Support Vector Machines*, recorreu-se ao *GridSearchCV* na tentativa de otimizar os hiperparâmetros *C*, *gamma* e *kernel*.

Datasets	Precision 0	Precision 1	Recall 0	Recall 1	f1_score 0	f1_score 1	Accuracy
D1	0.93	0.66	0.97	0.40	0.95	0.50	0.909
D2	0.92	0.66	0.97	0.45	0.94	0.53	0.901
D3	0.94	0.57	0.94	0.57	0.94	0.57	0.888
D4	0.91	0.64	0.99	0.21	0.95	0.32	0.899
D5	0.91	0.68	0.98	0.36	0.94	0.47	0.896

Figura 3.8: Resultados obtidos pelo algoritmo Support Vector Machines

XGBoost

Por fim, no algoritmo *XGBoost* utilizou-se, também, o *GridSearchCV*, de modo a otimizar os parâmetros *min_ child_ weight*, *gamma*, *subsample*, *colsample_ bytree* e *max_ depth*.

Datasets	Precision 0	Precision 1	Recall 0	Recall 1	f1_score 0	f1_score 1	Accuracy
D1	0.94	0.68	0.97	0.50	0.95	0.58	0.918
D2	0.93	0.66	0.96	0.53	0.95	0.59	0.906
D3	0.95	0.63	0.94	0.64	0.94	0.63	0.904
D4	0.91	0.67	0.99	0.21	0.95	0.33	0.901
D5	0.93	0.64	0.96	0.54	0.94	0.59	0.902

Figura 3.9: Resultados obtidos pelo algoritmo XGBoost

3.2 Deep Learning

Para a aplicação de modelos de *deep learning*, foi utilizado o *dataset* normalizado, onde foram removidos todos os valores nulos. Este *dataset* sofreu um *oversampling* da classe minoritária (classe 1) para os dados de treino, de forma a obter 5000 entradas. Da mesma forma, aplicou-se um *undersampling* aos dados de treino da classe 0, com o intuito de obter um conjunto de dados balanceado. Os dados de treino foram, posteriormente, divididos em conjuntos de treino e validação, permitindo o treino dos modelos de *deep learning*.

DNN

Foi implementada uma *Deep Neural Network* (DNN), onde foram criados sete modelos diferentes, fazendo variar entre eles o número de camadas densas, os filtros a usar em cada camada densa e a inclusão de *dropout*.

Esta implementação carece da otimização automática das redes, dificultando o processo de encontrar uma rede adequada para cada modelo. No *notebook* disponibilizado, são visíveis os resultados obtidos nas diferentes *epochs*, bem como o *evaluate* do modelo. Ademais, observa-se o modo como a *accuracy* no treino e na validação ao longo das *epochs* se relacionam, através de um gráfico. Nestas redes, a *accuracy* máxima obtida na validação e teste foi de 82%, sendo o mínimo de *loss* da validação 0.40.

LSTM

Posteriormente, foram implementadas redes *Long Short Term Memory* (LSTM). Para esta implementação, realizou-se um pré-processamento do *dataset* idêntico ao anterior.

Assim sendo, foram criados dois modelos distintos.

1. Modelo constituído por três camadas, duas camadas LSTM com 32 filtros e uma camada densa de *output*;
2. Modelo constituído por duas camadas, uma camada LSTM com 128 filtros e uma camada densa de *output*.

Estes modelos utilizam os mecanismos de *EarlyStopping* e *ModelCheckpoint* como *callbacks*. O *EarlyStopping* monitoriza a *val_ loss* e tem uma *patience* de 30. Por outro lado, o *ModelCheckpoint* guarda os pesos do modelo com menor *val_ loss*.

Com esta implementação, o melhor valor de *accuracy* obtido, no teste e validação, foi de 89%. O valor mínimo de *loss* para o conjunto de validação foi de 0.35, e para o teste de 0.34.

3.2.1 RNN

Na implementação das redes neuronais recorrentes, realizou-me um processamento do *dataset* idêntico aos anteriores. Para estas redes, foram implementados dois modelos.

1. Modelo constituído por uma camada *SimpleRNN*, com 32 filtros, e uma camada densa de *output*;
2. Modelo constituído por três camadas *SimpleRNN*, com 32 filtros, e uma camada densa de *output*.

Similarmente às redes LSTM, foram utilizados *EarlyStopping* e *ModelCheckpoint* como *callbacks* em ambos os modelos.

A melhor *accury* obtida foi de 84% nos dados de teste e 81% nos de validação. Quanto ao valor mínimo de *loss*, o seu valor foi de 0.37 para teste e 0.42 para validação.

3.3 Conclusões

Este projecto permitiu a consolidação dos vários métodos de aprendizagem máquina abordados na unidade curricular de Aprendizagem Automática II, tirando proveito da linguagem de programação Python e das suas bibliotecas.

No que diz respeito aos algoritmos de *Machine Learning* aplicados, a análise dos resultados obtidos permitiu verificar qual o melhor modelo aplicado a cada *dataset*. Desta forma, através da observação dos valores de *recall* e *accuracy*, verifica-se que o *XGBoost* é o modelo que obtém melhores resultados para o *dataset* 1. Para os *datasets* 2 e 4, o *Gradient Boosting* superou os resultados do *XGBoost*, sendo este o melhor modelo. Uma vez que os resultados para os *datasets* 3 e 5 foram idênticos, os melhores modelos aplicados são o *Gradient Boosting* e o *XGBoost*.

Em resposta às questões inicialmente colocadas, a observação da matriz de correlação revela que os atributos *duration*, *previous* e *poutcome* são os que mais favorecem a assinatura de um contrato. Os modelos preditivos que conseguem ter uma melhor percentagem de acerto, no que diz respeito a saber se assinou, ou não, contrato, foram o *Gradient Boosting* e o *XGBoost*, tendo este o valor mais alto de *accuracy*, com 91.8%. Por fim, a classe de maior dificuldade de previsão sem balanceamento é a classe minoritária (classe 1), devido ao elevado desbalanceamento apresentado pelo *dataset* quando em comparação com a classe 0.

Em suma, a realização deste trabalho exigiu a aplicação de todos os conhecimentos lecionados em contexto de aula, bem como a pesquisa de novos métodos, permitindo o desenvolvimento de modelos de *Machine* e *Deep Learning* capazes de produzir resultados satisfatórios para o problema em questão.