Predizendo Vendas em Telemarketing Bancário

Charles D. Moraes, Henrique T. Eihara, João Victor do C. Binatto, Rafael F. Zanetti
Departamento de Computação (DComp)
Universidade Federal de São Carlos (UFSCar)
18052-780, Sorocaba, São Paulo, Brasil
raverblazer@gmail.com, rick.eihara@gmail.com, joaovictorbinatto@hotmail.com, rfzanetti@gmail.com

***Resumo*—As empresas cada vez mais se utilizam de diferentes formas para agregar novos clientes à seus produtos. Uma das formas de atingir seus clientes é por campanhas de telemarketing, em que a prospecção de vendas se classficam como: *inbound* (quando o cliente tem a pró-atividade em comprar o produto), *outbound* (quando um intermediário oferece tal produto diretamente pro cliente). É extremamente importante identificar os possíveis clientes para que seja possível maximizar a prospecção de vendas do produto, dessa forma, utilizando de Sistemas de Suporte à Decisão em conjunto com algoritmos de Aprendizado de Máquina, é possível identificar com eficiência quem são os melhores clientes para se oferecer tal produto. Diante deste cenário, esse trabalho apresenta uma análise de técnicas de aprendizado de máquina aplicados na classificação dos melhores clientes. Utilizando uma base de dados real criada por um banco português, em sua campanha de marketing, foi possível realizar experimentos utilizando os algoritmos k-vizinhos mais próximos, regressão logística, redes neurais e máquina de vetores de suporte; resultados obtidos nos nossos experimentos indicaram que o uso de regressão logística é apropriado para construção de um modelo promissor na identificação dos usuários que tem uma maior chance de aderir à venda proporcionada por uma campanha de telemarketing.**

***Keywords*-telemarketing bancário; vendas; classificação.**

I. Introdução

As empresas cada vez mais se utilizam de diferentes formas para agregar novos clientes à seus produtos. Uma dessas formas é por meio de campanhas de telemarketing, forma que nasceu e cresceu na década 19 com a popularização do telefone. Com as empresas, cada vez mais com uma demanda maior para vender seus produtos, diversos empregos como Operador de Telemarketing foram criados, além de ser uma forma barata de campanha, é uma das formas em que mais clientes são atingidos.

Há duas classificações das prospecções de vendas de telemarketing, sendo elas, *inbound*, quando é o cliente que realiza a ligação para a empresa com interesse de comprar o produto, e *outbound*, quando o Operador de Telemarketing realiza a oferta de um produto para um cliente. Para ter êxito nas vendas do produto, é imprescindível conhecer o público alvo do produto e também o perfil do cliente que irá ser oferecido, dessa forma, escolher o melhor conjunto de clientes que poderão comprar o produto pode ser considerado parte do conjunto de problemas NP-Difícil [5].

Nesta situação, umas das soluções é a utilização de Sistemas de Suporte à Decisão, em que, são sistemas que utilizam um modelo genérico de tomada de decisão e tem como base sua análise em um grande conjunto de variáveis, dessa forma, através da análise dos dados é possível determinar um posicionamento e/ou escolha [7].

Muitos algoritmos de classificação são utilizados em Sistemas de Suporte à Decisão, sendo eles o k-vizinhos próximos, regressão logística, árvores de decisão, redes neurais, e ainda os mais recentes, Deep Learning (redes neurais) e máquinas de vetores de suporte que é o estado da arte na área de Aprendizado de Máquina. [6]

Neste artigo iremos avaliar o desempenho dos métodos de aprendizado de máquinas: k-vizinhos próximos, regressão logística, redes neurais e máquinas de vetores de suporte. Esses métodos poderão ser utilizados para aumentar a prospecção de venda de produtos, a partir da classificação do perfil dos clientes de uma campanha de telemarketing. O objetivo principal é realizar uma análise dos métodos de classificação em conjunto com os possíveis parâmetros promissores deste cenário.

Este artigo está estruturado da seguinte forma, na Seção II está apresentado a modelagem da base de dados, além de sua origem e características. Na Seção III é apresentado os métodos de aprendizado de máquina, seus parâmetros de configuração e suas características. Na Seção IV encontra-se os resultados obtidos a partir da execução desses métodos de classificação, e por último, na Seção V, são apresentadas as principais conclusões e análises deste trabalho.

II. Dados Utilizados

Nesta seção é apresentada a base de dados utilizada neste trabalho, juntamente com o processo de pré-processamento realizado para possibilitar o uso dos algoritmos desejados.

A base de dados original foi apresentada por [11] e contém dados de uma campanha de marketing de um banco de Portugal com 41.188 amostras com 21 atributos referentes às seguintes categorias:

- Dados do cliente contatado:
  - idade, profissão, estado civil, escolaridade, situação de inadimplência de crédito, existência de empréstimos pessoais e relacionados à moradia.
- Dados do último contato relacionado à campanha:

  - tipo do contato (telefone fixo ou móvel), mês, dia da semana e duração, em minutos.
- Dados de contexto social e econômico:
  - índice de desemprego (trimestral), índice de preços no consumidor (mensal), índice de confiança do consumidor (mensal), taxa interbancária oferecida em euro (diário) e número de empregados (trimestral).
- Outros:
  - numero de contatos com o cliente na campanha, número de dias do último contato relacionado a alguma campanha e resultado desta, e número de contatos com o cliente antes da campanha em questão.
- Resultado:
  - O sucesso ou não da campanha.

Após uma análise inicial dos dados, optou-se por remover o atributo relacionado à inadimplência, pois apenas três amostras no conjunto tinham valor positivo, e a maioria das amostras não tinha informação sobre seu valor.

Após essa remoção, a base ainda apresentava o problema de um número considerável de amostras com pelo menos um atributo faltando (aproximadamente 7% da base), o que prejudicava a aplicação dos algoritmos desejados. Pela disponibilidade de um número satisfatório de amostras, optou-se pela sua remoção, resultando em 38.247 amostras restantes.

O atributo relacionado ao número de dias desde o último contato foi notado como possível problema para os algoritmos, pois, nos dados, a representação de uma situação em que esse contato não existiu foi definida pelos seus criadores com o valor 999. Para os algoritmos, seria uma diferença extrema entre os outros valores, visto que o valor máximo apresentado na base é 33. Para lidar com isso, decidiu-se utilizar o valor 50 nessas situações, mantendo assim uma diferenciação razoável para casos em que esse contato existiu anteriormente e casos nos quais ele não existiu.

Outro problema encontrado foi a presença de atributos nominais, como por exemplo a profissão do cliente, o que complicava a utilização dos dados. Para tratá-lo, duas abordagens distintas foram tomadas:

- Para o atributo *escolaridade*, cujos valores nominais representam uma escala bem definida, os valores (*analfabeto, básico (4 anos), básico (6 anos), básico (9 anos), médio, superior, pós-graduação*) foram mapeados numa escala de 0 a 6.
- Cada um dos outros atributos nominais, cujo domínio não possibilita a abordagem acima, foi transformado em n atributos binários (sendo n o número de diferentes valores presentes). Por exemplo, o atributo *estado civil*, que apresentava os valores *solteiro, casado* e *divorciado*, foi mapeado em *bool-solteiro, bool-casado* e *bool-divorciado*. Exemplificando, valores de, respectivamente, *0, 1* e *0* nesses atributos são usados para representar o valor *casado* no atributo original.

Após o procedimento descito acima, totalizou-se 34 atributos na base. Analisando a distribuição de classes no conjunto de dados, restaram 33.988 amostras negativas e 4,258 amostras positivas. Em experimentos iniciais, verificou-se que, apesar de os algoritmos aplicados terem obtido bons níveis de acurácia, essa disparidade entre classes fazia com que a revocação da classe minoritária fosse baixa, ou seja, eles tendiam a classificar novas amostras na classe majoritária.

Para tratar esse problema, foram realizados testes com diferentes proporções de amostras positivas e negativas (com *undersampling*), avaliando sempre o MCC (*Matthews Correlation Coefficient*).

	Amostras positivas	Amostras negativas
Antes	4258	33988
Depois	4258	4258

Tabela I
DISTRIBUIÇÃO DE AMOSTRAS ANTES E DEPOIS DO BALANCEAMENTO

Os melhores resultados foram obtidos com o mesmo número de amostras nas duas classes, portanto decidiu-se pela remoção de amostras negativas aleatóriamente para ter uma base perfeitamente balanceada. A Tabela I apresenta a distribuição de classes antes e depois do processo de balanceamento.

III. METODOLOGIA EXPERIMENTAL

Foram definidos quatro algoritmos a serem utilizados para os experimentos de Aprendizado de Máquina: k-vizinhos mais próximos (*k-NN*) [4], regressão logística [9], redes neurais artificiais [10] e máquina de vetores de suporte (*SVM*) [3]. As próximas subseções reportam as metodologias experimentais gerais e específicas para cada algoritmo.

A. Implementação

Todos os algoritmos foram implementados utilizando *GNU Octave*.

B. Avaliação

Para avaliar a capacidade de generalização dos modelos construídos e evitar o super-ajustamento (*overfitting*), foi selecionada a técnica de validação cruzada com k partições, utilizando $k = 5$.

C. Métricas

As seguintes métricas foram escolhidas para avaliar o desempenho dos modelos construídos:

- Acurácia - porcentagem de amostras classificadas corretamente
- F-Medida - media harmônica entre precisão e revocação

- MCC (*Matthews Correlation Coefficient*) - medida de qualidade de classificações binárias

D. Normalização e Redução de Dimensionalidade

Uma técnica comumente utilizada para algoritmos de Aprendizado de Máquina é a normalização dos dados, para que o conjunto de valores de cada atributo tenha média 0 e desvio padrão 1, de modo que todos os atributos estejam na mesma escala.

Outra prática comum é a aplicação do PCA (Análise de Componentes Principais) para reduzir a dimensionalidade dos dados, com diferentes motivações (como por exemplo, a aceleração do treinamento).

Como os algoritmos a serem implementados tem propriedades bem distintas entre si, a aplicação das técnicas citadas acima foi avaliada separadamente para cada um. Essas avaliações, juntamente com metodologias especificas para cada algoritmo, são apresentadas nas subseções abaixo.

E. K-vizinhos mais próximos

Nos experimentos com o algoritmo k-vizinhos mais próximos, foram identificados problemas de acurácia devido à dimensionalidade dos dados. Devido a isso, o algoritmo apresentou problemas na classificação dos dados de teste, apresentando vários casos onde a distância de diversas amostras ficou idêntica, assim comprometendo o agrupamento dos k-vizinhos.

O cálculo da distância, inicialmente Euclidiana, foi modificado para *Manhattan* e posteriormente *Minkowski*, entretanto tais mudanças não refletiram numa melhora real na acurácia. Portanto, optou-se por continuar utilizando a distância Euclidiana.

Para tentar contornar os resultados insatisfatórios, foi utilizada a técnica do PCA (Análise das Componentes Principais) para reduzir a dimensionalidade da base, mantendo 95% de variância dos dados. Verificou-se uma melhora, porém não muito expressiva.

Finalmente, foram realizados testes com $K = 1, 3, 5, 7, 9$, e foi escolhido $K = 7$ por apresentar os melhores resultados.

F. Regressão Logística

Em experimentos preliminares, os resultados obtidos com a aplicação da Regressão Logística foram surpreendentemente eficazes. Porém, ao realizar testes com normalização dos dados, os resultados chegaram a melhorar ainda mais. Por isso, a normalização dos dados foi aplicada nos experimentos.

Com resultados satisfatórios e execução rápida, a redução de dimensionalidade não se mostrou necessária e não foi aplicada.

Para tentar aprimorar ainda mais os resultados, foi realizada uma busca pelo valor λ ótimo da regressão, sendo feita uma busca utilizando os valores 0, 0.01, 0, 05, 0.10, 0.25, 0.5, 0.75 e 1. O valor 0.01 apresentou os melhores resultados e foi selecionado para os experimentos.

G. Redes Neurais Artificiais

Em experimentos iniciais, os resultados indicaram uma classificação ruim e inconsistente, e uma análise mais profunda indicou que a otmização da função custo apresentava uma considerável demora para convergir. Um modo de corrigir esse problema é a normalização do conjunto de dados [13]. Testes com a aplicação dessa técnica apontaram resultados melhores e mais consistentes, portanto decidiu-se pela aplicação de normalização.

Foram realizados testes com redução de dimensionalidade, porém a diferença no tempo dos experimentos não foi grande o suficiente pra justificar a piora nos resultados devido à perda de variância dos dados.

Um dos parâmetros do algoritmo é a quantidade de iterações para otimização da função custo. A Figura 1 corresponde a uma curva que apresenta a relação entre valor da função custo e número de iterações:

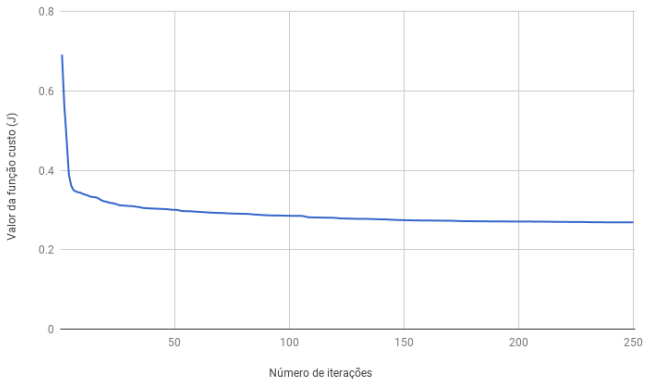

Figura 1. Curva de relação entre número de iterações e valor da função custo

De acordo com a curva, foi escolhido o valor de 75 iterações, por se tratar de um ponto onde a execução não é excessiva, porém com um nível satisfatório de otimização para a função custo.

Por fim, dois parâmetros inteferem diretamente no desempenho: o número de nós na camada intermediária, e o λ utilizado na regularização dos pesos. Para otimizá-los, foi realizada uma busca em *grid*, onde os seguintes valores foram testados, utilizando o MCC como referência para comparação:

- Número de nós: De 3 a 25.
- λ: $0, 0.01, 0, 05, 0.10, 0.25, 0.5, 0.75, 1$.

Após a busca, foram os selecionados 6 nós na camada intermediária e $\lambda = 0.05$.

H. Máquinas de vetores de suporte

Para implementar a técnica de máquinas de vetores de suporte utilizamos a conhecida biblioteca LIBSVM [1] e também uma breve leitura inicial de um guia [2], disponibilizado pelos principais responsáveis pela biblioteca.

Seguindo as recomendações providas pelo guia, iniciamos alguns testes com a função *kernel* Gaussiana (*Radial Basis Function*), dada por:

$$K(x, y) = e^{-\gamma||x-y||^2} \tag{1}$$

Portanto, temos 2 hiperparâmetros a serem escolhidos, C e γ. Sendo C o parâmetro de penalização, ou seja, é o *trade-off* entre classificar uma amostra de treino erroneamente contra a simplificação da superfície de decisão. Um valor baixo resulta em uma fronteira de decisão suave, e quanto maior o valor, maior a tendência de classificar todas as amostras de treino corretamente, dando ao modelo plena liberdade de escolher mais amostras como vetores de suporte.

Já o parâmetro γ define qual o alcance de influência de uma única amostra de treino, onde um valor baixo indica longo alcance e um valor alto indica curto alcance. Este parâmetro pode ser visto como o inverso do raio de influência das amostras selecionadas pelo modelo como vetores de suporte. É possível observar, de forma mais intuitiva e visual o comportamento destes parâmetros na Figura 2.

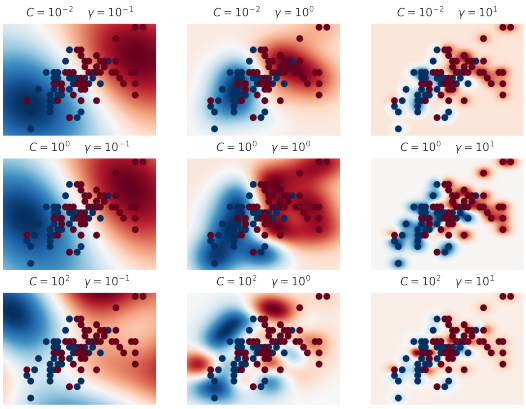

Figura 2. Demonstração do comportamento de C e γ utilizando como exemplo o *Iris Dataset* [14]

Após termos escolhido inicialmente o *kernel* RBF para os primeiros testes, uma leitura mais a fundo mostrou também os seguintes pontos importantes que nos levaram a escolha definitiva desta função *kernel*.

O *kernel* RBF mapeia as amostras de forma não linear para um espaço dimensional maior, portanto, pode lidar com casos onde a relação entre os atributos e a classe não são lineares. Além do mais, ele possui a mesma performance de um *kernel* linear com parâmetro de penalização $\tilde{C}$ [12].

Outro fato muito importante é o número de hiperparâmetros a serem escolhidos, os quais influenciam a complexidade de seleção do modelo. O *kernel* polinomial é dado por:

$$K(x, y) = (\gamma x^T y + r)^d \tag{2}$$

Ou seja, temos quatro hiperparâmetros para serem selecionados: C, γ, r e d. Além disso, o *kernel* polinomial pode sofrer com dificuldades numéricas, onde os valores do *kernel* podem tender a infinito ($\gamma x^T y + r > 1$) ou zero ($\gamma x^T y + r < 0$) quando o grau(d) é alto, enquanto os valores do *kernel* RBF ficam entre 0 e 1 ($0 < K_{ij} < 1$).

Para otimizarmos os hiperparâmetros C e γ realizamos busca em *grid* com *5-fold cross-validation*. Segundo o guia, sequências que crescem exponencialmente são um método prático para se identificar bons parâmetros [2]. Portanto, seguindo as recomendações do mesmo, optamos por realizar duas buscas em *grid*. Ambas as buscas foram realizadas duas vezes, sendo uma aplicando PCA e outra não, e ambas com normalização dos atributos.

A primeira busca possui um intervalo mais amplo entre os valores exponenciais, sendo seu *step* de 2^2 e as seguintes sequências: $C = 2^{-5}, 2^{-3}, ..., 2^{15}$ e $\gamma = 2^{-13}, 2^{-11}, ..., 2^{1}$. Após realizada a primeira busca agrupamos todos os resultados por cada parâmetro e tiramos a média e desvio padrão da acurácia e MCC para identificamos os intervalos de C e γ que obtiveram os melhores resultados, como podemos obserar na Tabela II.

	C		γ		
	Acc	MCC	Acc	MCC	
2^{-5}	0.65 ± 0.14	0.32 ± 0.28	0.80 ± 0.12	0.60 ± 0.23	2^{-13}
2^{-3}	0.73 ± 0.13	0.49 ± 0.24	0.84 ± 0.06	0.68 ± 0.12	2^{-11}
2^{-1}	0.79 ± 0.09	0.60 ± 0.15	$\mathbf{0.85 \pm 0.04}$	$\mathbf{0.70 \pm 0.08}$	2^{-9}
2^{1}	$\mathbf{0.82 \pm 0.06}$	$\mathbf{0.66 \pm 0.10}$	$\mathbf{0.85 \pm 0.03}$	$\mathbf{0.70 \pm 0.05}$	2^{-7}
2^{3}	$\mathbf{0.83 \pm 0.06}$	$\mathbf{0.66 \pm 0.11}$	$\mathbf{0.84 \pm 0.03}$	$\mathbf{0.68 \pm 0.05}$	2^{-5}
2^{5}	$\mathbf{0.83 \pm 0.06}$	$\mathbf{0.66 \pm 0.11}$	0.81 ± 0.04	0.62 ± 0.06	2^{-3}
2^{7}	0.82 ± 0.06	0.65 ± 0.11	0.72 ± 0.09	0.47 ± 0.18	2^{-1}
2^{9}	0.82 ± 0.06	0.65 ± 0.11	0.65 ± 0.09	0.34 ± 0.19	2^{1}
2^{11}	0.82 ± 0.06	0.64 ± 0.11	-	-	-
2^{13}	0.81 ± 0.06	0.63 ± 0.11	-	-	-

Tabela II
RESULTADOS DO *Coarse Grid Search* SEM PCA

Apesar da sequência que definimos inicialmente para C ir até 2^{15} não chegamos a concluir a busca em *grid* neste último exponencial devido ao alto custo computacional, exigindo um grande número de iterações pela SVM e demorando consideravelmente em cada *fold* da validação cruzada, além disso, tanto a acurácia quanto o MCC já se encontravam em queda a partir do exponencial 2^7.

A segunda busca possui um intervalo mais curto entre os valores exponenciais, sendo seu step de $2^{0.25}$ e as sequências obtidas a partir da primeira busca foram: $C = 2^1, 2^{1.25}, ..., 2^5$ e $\gamma = 2^{-9}, 2^{-9.25}, ..., 2^{-5}$, como pudemos observar na Tabela II.

Feita essa busca em *grid* mais fina, selecionamos $C = 2^{4.25}$ e $\gamma = 2^{-7.25}$ sem PCA, como os hiperparâmetros ótimos.

IV. Resultados

Nesta seção, serão apresentados os resultados da aplicação dos algoritmos anteriormente citados na predição de vendas em telemarketing bancário, juntamente com sua discussão.

Foram realizados dez experimentos com cada algoritmo, e a Tabela III apresenta a média e o desvio padrão dos resultados obtidos, ordenadas por MCC. Os valores em negrito representam os melhores desempenhos para cada métrica avaliada.

Teste	Acc (%)	F-Medida	MCC
Reg. Logística	$\mathbf{97.54 \pm 0.12}$	$\mathbf{0.976 \pm 0.012}$	$\mathbf{0.952 \pm 0.023}$
SVM	87.55 ± 0.14	0.880 ± 0.014	0.753 ± 0.029
Redes Neurais	87.47 ± 0.10	0.879 ± 0.012	0.751 ± 0.022
k-NN	51.86 ± 1.28	0.540 ± 0.013	0.038 ± 0.026

Tabela III
Resultados

Analisando os resultados, vemos uma clara disparidade entre o k-vizinhos e os outros algoritmos. Os resultados do k-vizinhos indicam uma classificação levemente melhor do que a aleatoriedade, fazendo com que não seja uma boa opção para prever as vendas utilizando a base que obtemos.

Em [8] é verificado que, após as primeiras 20 dimensões de atributos, a distinção da distância entre as amostras diminui-se num nível extremamente rápido, fazendo com que bases excedendo esse número de atributos, como é o caso, tenham resultados ruins. Em contrapartida, a aplicação de redução de dimensionalidade para fugir desse problema faz com que a base perca uma alta porcentagem de variância, sendo esse o culpado dos resultados ruins aplicando o PCA.

Podemos categorizar os resultados dos outros algoritmos em dois grupos:

- *SVM e Redes Neurais*: Apresentaram resultados extremamente similares, com o SVM tendo uma minuscula vantagem. Ambos tiveram acurácia de mais de 87%, indicando-os como uma alternativa viável para a predição do sucesso ou não das campanhas de telemarketing. O valor obtido do MCC, em torno de 0.75, indica uma correlação boa para os modelos treinados.

  Vale notar que o balanceamento de dados citado na Seção II foi um fator de extrema importância para a obtenção deste resultado, pois os dados desbalanceados, apesar de fazerem com que os modelos apresentassem níveis de acurácia superiores aos reportados nesta seção, o valor do MCC era bem baixo, pois a maioria das amostras de teste eram classificadas como sendo da classe majoritária. Foi definido que o valor do MCC representa uma métrica mais representativa da qualidade do modelo para o cenário deste trabalho.
- *Regressão Logística*: Apresentou, com uma boa vantagem, os melhores resultados em todas as métricas. Além de tudo, foi o método que apresentou execução mais rápida. Com acurácia de 97.5%, F-medida de 0.976 e 0.952 de MCC, o modelo treinado indica ser, com certeza, o melhor para a previsão, neste cenário, das vendas ou não para os clientes.

V. Conclusões

Este trabalho apresentou os resultados obtidos de quatro métodos de classificação aplicados ao *Bank Marketing Data Set*[11] após várias análises e aplicações de técnicas de Aprendizado de Máquina.

O classificador *kNN*, além de computacionalmente custoso não obteve resultados satisfatórios, obtendo uma acurácia total de 51%, já os classificadores de Redes Neurais e SVM obtiveram bons resultados, acurácia total em ambos 87%, aliados à um bom desempenho computacional, sendo custoso apenas a otimização dos parâmetros, exigindo uma busca em *grid* extensiva. Por fim, o melhor classificador foi o de Regressão Logística, com uma acurácia de 97.54%. Por ser uma técnica de baixa complexidade computacional é uma excelente escolha para este conjunto de dados dada a sua alta precisão de classificação.

Trabalhos futuros podem incluir análises focadas em identificar os principais fatores que levam um cliente aderir ou não à campanha de telemarketing, assim fazendo com que os bancos saibam os pontos exatos onde devem investir, visando aumentar a participação de clientes, e, consequentemente, seus lucros.

A realização desse trabalho envolveu a aplicação de inúmeros conceitos aprendidos na disciplina de Aprendizado de Máquina e forçou os autores a fortificar seus conhecimentos e verificar, na prática, as várias variáveis incluídas no processo de desenvolver um modelo preditivo de dados.

Referências

[1] C. Chang and C. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[2] C. Hsu, C. Chang and C. Lin. A practical guide to support vector classification, 2010. Available at http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf.

[3] C. Cortes and V. Vapnik. *Support-vector networks*, Machine Learning 20:273-297, 1995

[4] D. W. Aha, D. Kibler and M. K. Albert. *Instance-based learning algorithms*, Machine Learning 6:37-66, 1991

[5] F. Nobibon, R. Leus and F. Spieksma. *Optimization models for targeted offers in direct marketing: exact and heuristic algorithmns*, European Journal of Operational Research 210, 670:683, 2011

[6] J. Diederich. *Rule Extraction from Support Vector Machines*, Springer, 34, 2008

[7] J. O'Brien and G. Marakas *Administração de Sistemas de Informação*. Porto Alegre: AMGH. p. 9., 2013

[8] K. Beyer, J. Goldstein, R. Ramakrishnan and U. Shaft. *When Is "Nearest Neighbor" Meaningful?*, Springer-Verlag, 17, 1998

[9] S. Le Cesse and J. C. Van Houwelingen. *Ridge estimators in logistic regression*, Applied Statistics:191-201, 1992

[10] S. Haykin. *Neural networks: a comprehensive foundation*, Tsinghua University Press, 2001

[11] S. Moro, P. Cortez and P. Rita. *A Data-Driven Approach to Predict the Success of Bank Telemarketing*, Decision Support Systems, Elsevier, 62:22-31, 2014

[12] S. S. Keerthi and C. Lin. Asymptotic behaviors of support vector machines with Gaussian kernel. *Neural Computation,* 15(7):1667–1689, 2003.

[13] Y. LeCun et al. "Efficient backprop."Neural networks: Tricks of the trade. Springer Berlin Heidelberg, 2012. 9-48.

[14] Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

APÊNDICE

Estrutura de diretórios da implementação:

- `algorithms`
  - `common`
  - `knn`
  - `logistic_regression`
  - `neural_networks`
  - `svm`
- `data`
- `pre_processing`

O diretório `data` contém os arquivos de dados `raw_data`, com os dados originais da base, `data`, com os dados pré-processados, e `balanced_data`, com os dados pré-processados e as classes balanceadas.

O diretório `pre_processing` contém os scripts de pré-processamento dos dados. Devem ser executados em ordem, de 1 à 3. Sua execução é feita da seguinte maneira: `python script.py`, sendo `script.py` um dos 3 scripts.

O diretório `algorithms` contém sub-diretórios para as quatro técnicas implementadas neste trabalho, além de um diretório `common`, que possui funções comuns que são utilizadas por mais de um algoritmo, foi neste diretório que armazenamos as funções: `accuracy`, `f_measure`, `mcc`, `normalizar`, `pca` e `projetarDados` além de um *template* como base para a validação cruzada.

Para criação dos resultados demonstrados no artigo, nos algoritmos k-vizinhos mais próximos e regressão logística, devemos executar um script chamado `start.sh` em ambos algoritmos, esse script realizará todos os possíveis testes e também deixara cada resultado em uma pasta chamada resultado concatenado com a data do ínicio do teste. Para os algoritmos de redes neurais e máquina de suporte de vetor, pode se executar as rotinas de busca em *grid* pelo GNU Octave, sendo elas respectivamente: `neural_networks_experiment.m` e `grid.m`

Para executar os classificadores nas configurações ótimas, apenas execute pelo GNU Octave, o arquivo `main.m` que se encontra nos algoritmos k-vizinhos mais próximos, regressão logística e máquina de suporte de vetor. Para as redes neurais pode ser o mesmo arquivo citado anteriormente: `neural_networks_experiment.m`.