UNIVERSIDADE
FEDERAL RURAL
DE PERNAMBUCO

Bruno Roberto Florentino da Silva

Aprendizagem de máquina para a identificação de clientes propensos à compra em *Inbound* marketing

Recife

Julho de 2019

Bruno Roberto Florentino da Silva

Aprendizagem de máquina para a identificação de clientes propensos à compra em *Inbound* marketing

Monografia apresentada ao Curso de Bacharelado em Sistemas de Informação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Universidade Federal Rural de Pernambuco

Departamento de Estatística e Informática

Bacharelado em Sistemas de Informação

Orientador: Cleviton Monteiro

Coorientador: Rodrigo Soares

Recife

Julho de 2019

Dados Internacionais de Catalogação na Publicação
Universidade Federal Rural de Pernambuco
Sistema Integrado de Bibliotecas
Gerada automaticamente, mediante os dados fornecidos pelo(a) autor(a)

D111a Da Silva, Bruno Roberto Florentino
Aprendizagem de máquina para a identificação de clientes propensos à compra em inbound marketing / Bruno Roberto Florentino Da Silva. - 2019.
88 f. : il.

Orientador: Cleviton Vinicius Fonseca Monteiro.
Coorientador: Rodrigo Gabriel Ferreira Soares.
Inclui referências, apêndice(s) e anexo(s).

Trabalho de Conclusão de Curso (Graduação) - Universidade Federal Rural de Pernambuco, Bacharelado em Sistemas da Informação, Recife, 2023.

1. Marketing digital. 2. leads. 3. aprendizado de máquina. 4. classificação supervisionada. 5. desbalanceamento dos dados. I. Monteiro, Cleviton Vinicius Fonseca, orient. II. Soares, Rodrigo Gabriel Ferreira, coorient. III. Título

CDD 004

Bruno Roberto Florentino da Silva

APRENDIZAGEM DE MÁQUINA PARA A IDENTIFICAÇÃO DE CLIENTES PROPENSOS À COMPRA EM *INBOUND* MARKETING

Monografia apresentada ao Curso de Bacharelado em Sistemas de Informação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Aprovada em: 12 de julho de 2019.

BANCA EXAMINADORA

Cleviton Monteiro (Orientador)
Departamento de Estatística e Informática
Universidade Federal Rural de Pernambuco

Rodrigo Soares
Departamento de Estatística e Informática
Universidade Federal Rural de Pernambuco

Silvana Bocanegra
Departamento de Estatística e Informática
Universidade Federal Rural de Pernambuco

A Nosso Senhor Jesus Cristo, à minha Mãezinha de Lourdes, à minha amada família, à minha "mar bunita", aos amigos queridos e todos aqueles que sofrem de síndrome de pânico e transtornos de ansiedade.

"No mundo, o êxito só é possível com um desenvolvimento contínuo. Não progredir é cair."

(Antônio Frederico Ozanam)

Agradecimentos

Começo agradecendo a quem tudo me proporcionou, meu Senhor, meu Deus. A Nossa Senhora que com certeza intercedeu por mim durante esta caminhada, assim como São Vicente de Paulo e o Beato Ozanam, que com certeza estavam comigo cada vez que eu rogava pedindo para conseguir chegar na faculdade.

Agradeço a minha família, em especial a minha mãe Jardilene e meu pai José Humberto que serviram de base para que eu chegasse até aqui. As minhas tias Zildete e Iaiá pelas orações e apoio por toda minha vida. Agradeço também a meu tio Davi, por sempre estar disposto a me ajudar e ir comigo até a universidade nos dias que não conseguia ir sozinho. Agradeço também a meus irmãos de criação, João Vitor e Rafaela Andrade por todo amor doado.

Agradeço aos amigos que fiz durante a graduação, cada um sabe a importância que teve para mim nessa jornada. A Leonardo Pontes pelo companheirismo, amizade e apoio para que eu conseguisse subir as escadas e assistir as aulas, mesmo estando com crises de ansiedade; assim como, pelas calouradas e farras em que nos divertimos juntos. A José Clodoalves pela confiança, amizade e suporte nos momentos que precisei de sua ajuda. A Raffael Vieira pela amizade ímpar e principalmente pelas “Bizuradas’’. A meu amigo Daniel Cândido pelo apoio fundamental, como também pelas tardes de futebol, piscina, cerveja e churrasco. A Rodolfo Bispo pelo apoio e disposição em querer sempre ajudar. A Robson César pela confiança, solidariedade e pelos bons momentos. A Demis Gomes, Jorge Delgado, Júnior Costa, Delando Junior, Vinicius Acioly, Waleska Dias e Kelvin Souza, pelo apoio e suporte sempre que precisei.

Ao grande amigo que a Editora UFPE me concebeu, Gabriel Santana, por me ajudar a fazer coisas que não conseguia fazer sozinho, desde ir no Centro, até ir para o jogo do Sport, ou cinema. Assim como, a meu amigo Luiz Reis pela amizade ímpar, pelas tardes de PES, churros e muita Coca-Cola. Agradeço a todos os “bobocas”, com quem ri, me diverti e aprendi muito.

A meus amigos de longa data Edson Oliveira, André Alves, Júnior Chacon, Rodrigo Cristóvão, Renan Barbosa, Ricardo Medeiros e Palloma Tyane, pela amizade verdadeira, pela paciência e pelo apoio não só durante a graduação, como também durante toda a vida.

Agradeço em especial a minha namorada Pâmella Tuane pela paciência, amizade e companheirismo, estando do meu lado em dias bons, assim como nos dias que eu não conseguia ir para faculdade sozinho e também pela renúncia de suas vontades estando sempre do meu lado durante a realização deste trabalho.

Agradeço a meu orientador Cleviton Monteiro pelo suporte, paciência, confiança e por não ter desistido de mim. A meu coorientador Rodrigo Soares por acreditar em mim em cada momento de elaboração deste trabalho e por me motivar a querer buscar sempre o melhor que a Universidade pode proporcionar. Aos professores Gabriel Alves, Glauco Gonçalves e Vitor Medeiros, pelos grandes ensinamentos passados. A Jones Albuquerque por toda a inspiração. À Silvana Bocanegra, pela compreensão quando coordenadora do curso, de sempre alocar as disciplinas que eu ia cursar nas salas do térreo, porque eu não conseguia subir o prédio. Um agradecimento especial também a melhor e mais simpática secretária que já conheci, Carol.

Por fim, peço perdão a todos aqueles que esqueci de mencionar e agradeço-os de coração. A todos vocês, meu muito obrigado! Minha vida ficou melhor com vocês!

Resumo

O ponto mais importante para uma empresa deve ser sempre o cliente e, conseguir novos clientes, nem sempre é uma estratégia fácil. As técnicas de marketing digital estudam como atrair novos clientes para as empresas fazendo uso de plataformas digitais. Em virtude da popularização destes meios, as estratégias tiveram que se moldar às novas necessidades. Com apenas um clique é possível alcançar milhares de indivíduos, o que significa muitos leads (oportunidades de negócio) novos para a empresa. Entretanto, filtrar quais desses indivíduos estão realmente interessados no produto ou serviço ofertado pela empresa demanda um grande esforço da equipe de vendas. Essa sobrecarga é prejudicial no sentido de que a empresa pode perder receita por falta de direcionamento das verdadeiras oportunidades. Visando amenizar tal problema, o presente trabalho oferece uma proposta cujo objetivo é a identificação automática de potenciais clientes com maior propensão à compra dentre os leads obtidos por uma empresa através de estratégias de marketing digital.

Para tornar possível a execução desta proposta, foram utilizados recursos de Aprendizado de máquina, com aplicação dos algoritmos de classificação supervisionada, Árvore de decisão e Naive Bayes (NB), fornecidos pela biblioteca Scikit-learn, sob a linguagem de programação Python. Além disso, fez-se necessário a aplicação do algoritmo de sobreamostragem SMOTE, devido ao desbalanceamento do conjunto de dados. Com a finalidade de otimizar a classificação, foram utilizadas técnicas de seleção de atributos e seleção de modelos com ajuste de hiperparâmetros. Para avaliação dos resultados, utilizou-se as métricas de matriz de confusão, precisão, cobertura e curva de precisão e cobertura.

Devido ao desbalanceamento dos dados, a métrica de precisão não relatou bons resultados, com médias de 5,5% de acerto. Já a cobertura alcançou médias de aproximadamente 83%. Mesmo com resultados tão divergentes entre as métricas aplicadas, o presente trabalho conseguiu identificar a maioria das verdadeiras oportunidades e relatando que ao utilizar esta abordagem, seria possível obter uma redução de até 85% da aplicação de esforço por parte da equipe de vendas. Em consequência disso, uma empresa pode ter uma redução de custos ao diminuir os

recursos aplicados para obter novos clientes, propiciando que a equipe de vendas possa encontrar novos clientes com maior eficiência.

Palavras-chave: Marketing digital, *leads*, aprendizado de máquina, classificação supervisionada, desbalanceamento dos dados.

Abstract

The most important point for a company should always be the customer and getting new customers is not always an easy strategy. Digital marketing techniques study how to attract new customers to businesses using digital platforms. By virtue of the popularization of these means, the strategies had to be shaped to the new possibilities. With just one click you can reach thousands of individuals, which means many new leads for the company. However, filtering out which of these individuals are really interested in the product or service offered by the company demands a lot of effort from the sales team. This overhead is detrimental in the sense that the company can lose revenue by not targeting the real opportunities. With the aim to minimize this problem, the present work offers a proposal whose objective is the automatic identification of the client achieved through digital marketing strategies.

It is proposed the usage of Machine Learning techniques, in particular supervised classification algorithms, namely Decision Tree and Naive Bayes. It was used the Scikit-learn library available for the Python programming language. In addition, it was necessary to apply the SMOTE oversampling algorithm, due to the unbalance of the dataset. In addition, in order to optimize the classification, we used the techniques of attribute selection and model selection with hyperparameters adjustment. Finally, to evaluate the results, we used the confusion matrix, the precision and coverage metrics, and the accuracy and coverage curve.

Due to the imbalance of the data, the precision metric did not report good indexes results, with averages of 5.5% of correctness. In addition, the coverage was around 83%. Even with such divergent results among the applied metrics, the present work reached its goal, identifying most of the real opportunities and reporting that using this approach, it would be possible to obtain a reduction of up to 85% in the effort applied by the sales team if they had to call for all the leads. As a consequence, the company may have a cost reduction with the resources applied to obtain new customers, allowing the sales team to find new customers with greater efficiency.

Keywords: Digital marketing, leads, machine learning, supervised classification, data imbalance.

Lista de ilustrações

Lista de Tabelas

Lista de abreviaturas e siglas

SMOTE	Synthetic Minority Over-sampling Technique
CART	Classification and Regression Trees
TP	True positive
FP	False positive
FN	False negative
TN	True negative
AS	Seleção de atributos

Sumário

1 Introdução

1.1 Apresentação e motivação

A popularização do mundo digital causou grande impacto no comportamento dos consumidores e produtores e, por conseguinte, nas estratégias de marketing. O surgimento dos canais digitais fez com que métodos já consagrados fossem reavaliados e reinventados de maneira a suprir tais mudanças: o que antes se tratava de uma divulgação atrativa da marca feita de maneira unilateral, passou a ser uma relação de duas vias entre a empresa e o cliente (KOTLER, 2010).

> O consumidor é o mesmo, e seu comportamento online reflete os desejos e valores que ele traz de sua experiência na sociedade [...] O que a internet fez de fato, foi abrir de novo as portas para a individualidade e para o coletivo, sem a mediação de nenhum grupo de interesse (TORRES, 2009).

Com isso, o marketing passou a usar a internet como um de seus principais canais de venda, informação e conteúdo, o que possibilitou um alcance muito maior de prováveis consumidores, que eram antes limitados a regiões específicas. Surge então o Marketing digital, com mesma a finalidade proposta por Kotler: satisfazer as necessidades de lucratividade.

Segundo VAZ (2010), destacam-se as empresas que mantém um diálogo mais próximo com seus clientes, pois conseguem mais facilmente apresentar o que eles procuram, seja em promoções, serviços e vantagens, ficando à frente dos concorrentes. Isso foi possível com a chegada do *Inbound* Marketing, que engloba estratégias de marketing que exploram mecanismos de busca, blogs e redes sociais, só que com uma diferença para o marketing tradicional: nesse caso é o cliente que procura a empresa e não o contrário. Para isso, são realizadas ações no intuito de atrair leads: o blog, site, ou rede social da empresa oferece conteúdo personalizado em troca de informações do potencial cliente (nome, e-mail, telefone, cargo, por exemplo) (RESULTADOS DIGITAIS, 2016).

Empresas especializadas em *Inbound* Marketing, possuem softwares automatizados, que, ao oferecer material atrativo, criam bases de dados com leads:

potenciais clientes que são encontrados através de suas “pegadas digitais”. Funciona da seguinte forma: uma vez que alguém acessa o portal de uma empresa que disponibiliza conteúdo, é porque existe interesse no assunto, seja para contratar um serviço, seja para estudo e ampliação do conhecimento, ou simplesmente por navegação aleatória. O software oferece então, conteúdo extra, através de vídeos, webinars, folders, e-books, entre outros. Para obter acesso ao conteúdo é preciso que, em troca, o indivíduo deixe informações como e-mail, número de telefone, sua localização, seu cargo na empresa no qual faz parte, entre outras, que serão todas armazenadas numa enorme base de dados. Isso cria um desafio para a empresa: a base de dados conterá informações de todos os indivíduos que mostraram interesse no conteúdo oferecido. O problema é que a maioria deles não será um possível novo cliente, ou seja, a equipe de vendas da empresa terá um trabalho enorme e de alto custo para selecionar manualmente em quais leads devem investir tempo, além de requerer uma boa experiência dos membros da equipe, inclusive para não deixar passar possíveis oportunidades.

Uma resposta para este problema pode estar na seleção de leads quentes (potenciais clientes), fazendo uso de técnicas de Reconhecimento de Padrões. Trata-se de um campo da ciência que se preocupa com a descoberta automática de regularidades em dados através do uso de algoritmos de computador (BISHOP, 2006). Com isso, é possível construir uma representação mais simples do conjunto de dados através de suas características mais relevantes, possibilitando sua partição em classe (DUDA et al., 2001). O presente trabalho propõe que a seleção de leads seja feita de forma automatizada através da execução de algoritmos clássicos de Aprendizado Supervisionado de Máquina, já consagrados no Reconhecimento de padrões. Em síntese, esses algoritmos de Inteligência Artificial conseguem aprender com a base de dados, identificando padrões nos dados que podem ser procurados automaticamente, detectados, validados e usados para previsão (WITTEN et al., 2005), então será possível prever quais indivíduos serão, de fato, potenciais clientes e quais não são interessantes para a equipe de vendas.

1.2 Objetivos

O presente trabalho tem como objetivo principal à identificação automática de potenciais clientes com maior propensão à compra dentre os leads obtidos por uma

empresa através de estratégias de marketing digital. Para atingir esse objetivo geral, os seguintes objetivos específicos foram delimitados:

- Aplicar os algoritmos de aprendizado de máquina Árvore de decisão e Naive Bayes para fazer a classificação supervisionada de um conjunto de dados de duas classes (binário).
- Preparar o conjunto de dados e realizar seu pré-processamento.
- Fazer a seleção dos atributos relevantes para a classificação.
- Fazer o ajuste de parâmetros utilizando a abordagem de pesquisa aleatória junto ao método de validação cruzada aninhada.
- Analisar os resultados encontrados, considerando as métricas matriz de confusão, precisão e cobertura.
- Analisar a possibilidade de melhorar os resultados e comparar com trabalhos relacionados.

1.3 Organização do trabalho

Este trabalho encontra-se dividido em 5 Capítulos. No Capítulo 2 será apresentado o referencial teórico que conterá todos métodos que embasaram o desenvolvimento do trabalho. No Capítulo 3 será descrito detalhadamente como foram realizados os experimentos. No Capítulo 4, serão mostrados os resultados obtidos, assim como sua análise e comparação com trabalhos relacionados. No Capítulo 5, será a apresentada a Conclusão e considerações finais do trabalho.

2 Referencial teórico

2.1 Marketing Digital

Em uma organização, o marketing funciona com uma ideologia, tem valor cultural e sua necessidade está ligada aos anseios e desejos dos consumidores. Para KOTLER (1996), marketing é uma atividade direcionada para a satisfação das necessidades e desejos humanos, por meio de um processo de troca. Ou seja, é uma estratégia de valor imensurável já que as necessidades dos clientes estão em constante mudança, devido às novidades que vão surgindo. Assim, tem como dever ganhar destaque nos processos e tendências dos cenários competitivos das empresas. (TOLEDO et al., 2006).

Nos anos 2000, quando computadores passaram a ser finalmente acessíveis para o consumidor final, a internet também popularizou-se dando início a chamada Revolução Digital. Em virtude disso, tornou-se necessário uma adaptação dos ambientes organizacionais que foram se moldando a essa nova realidade para conseguir sobreviver em meio ao novo cenário. GARCIA (2007), afirma que as pessoas que utilizam a internet diariamente, a avaliam como favorável e importante por agregar informações e poder ser utilizada como uma ferramenta de compra.

Com isso, o conceito de marketing modificou-se substancialmente pois as pessoas passaram a procurar conteúdo relevante na internet, assim como passou a interagir com a instituição que divulgou tal conteúdo. Nesse contexto, é possível identificar uma mudança no comportamento dos consumidores que passaram a preocupar-se mais com os valores de uma organização, surgindo o Marketing 3.0 (KOTLER, 2010). O referido autor, assegura que essa é a era "da participação", "do paradoxo da globalização" e da "sociedade criativa" e deve ser entendida como ambiente propício para adoção de um "marketing colaborativo, cultural e espiritual".

Esse ambiente de colaboração passou a ser facilmente observado nas mídias sociais, onde é possível ver uma relação mais expressiva entre o cliente e a empresa. Os consumidores passaram a ser “prosumidores”, pois agora são consumidores que produzem conteúdo, em qualquer meio disponível na web (KOTLER, 2010). E foi nesse contexto que surgiu o marketing digital, que é um conjunto de estratégias de

marketing, inclinadas a comunicação, com fim comercial, entre a empresa e o cliente, possibilitada pela internet e aplicativos (SOUZA, 2012). Essas estratégias viabilizaram novos caminhos para divulgação de produtos e serviços, abrindo oportunidades para a conquista de mais clientes, bem como aumentou a rede de relacionamentos entre as empresas e os clientes (SEGURA, 2009).

A mudança do consumo para o mundo digital, a simplicidade da maneira de buscar pelos melhores produtos e serviços e o desejo pelo consumo feito em tempo real, mudou totalmente a visão das empresas, que foram obrigadas a tornar-se mais exigentes em relação às novas estratégias, pois o seu perfil está em constante atualização. Com isso, o marketing digital tornara-se importante na percepção do processo de decisão do consumidor. Muitas vezes ele é o responsável por gerar as sensações, desejos e até o reconhecimento do problema que poderá influenciar o indivíduo a realizar a compra, inclusive também o comportamento de pós-venda, que pode tornar o indivíduo fiel a marca.

2.2 *Inbound* marketing e o Marketing de conteúdo

Implementar estratégias de marketing digital com eficiência, é fundamental para delimitar e instigar seguidores e consumidores (CARO, 2010). Com a facilidade de interação entre as empresas, clientes e seus demais públicos, algumas estratégias do marketing digital têm custo quase zero quando comparadas com as do marketing tradicional (SOUZA, 2012). Para SOLOMON (2011), este estilo de marketing detém uma maior capacidade de divisão, praticidade e comunicação personalizada, o que o torna mais econômico em contraposto das ações de comunicação convencional.

Uma das estratégias do marketing digital é o *Inbound* marketing. Na tradução direta, o marketing de entrada se caracteriza por atrair potenciais clientes através dos meios digitais, oferecendo conteúdo relacionado a um determinado produto ou serviço oferecido pela empresa, em troca dos dados das pessoas interessadas. Na Figura 1 podemos observar um resumo das fases do *Inbound* marketing.

Figura 1 - Metodologia do *Inbound* marketing.
(COELHO MARKETING, 2019)

Primeiro, é necessário atrair pessoas diversas, seja através de blogs, redes sociais, dentre outras ferramentas, através de conteúdo relevante, é o chamado marketing de conteúdo. Cada etapa usa algum tipo específico de marketing de conteúdo que será oportuno para o momento, por exemplo, o assunto tratado com visitantes será diferente do que será enviado a clientes que já compraram o produto. Após o primeiro contato, o visitante buscará o conteúdo que lhe interessa, que será disponibilizado através da primeira troca de informações com a empresa. Existem empresas especializadas em lidar com *Inbound* marketing, que possuem softwares inteligentes capazes de armazenar todas as informações obtidas nessa troca. Essas empresas são contratadas por uma organização com intuito de ajudar a equipe de marketing a converter (transformar os indivíduos que deram suas informações - leads), em possíveis oportunidades para a empresa. As informações são então armazenadas numa base de dados, que será trabalhada pela equipe de marketing que por sua vez, deve traçar um perfil padrão para um lead ser considerado uma boa oportunidade. Os visitantes que foram convertidos em leads, precisam de atenção específica e quando recebem o conteúdo correto, ficam mais propensos a realizar a compra. E-mails de marketing são fundamentais nessa etapa, pois uma vez que um visitante se tornou um lead, a organização não pode esperar que ele procure novamente por mais conteúdo. É possível que ele tenha encontrado outra solução. Portanto, alimentar essa relação é fundamental. É preciso tomar a iniciativa de trabalhar o marketing de conteúdo, ou seja, enviar algo novo, que seja interesse do lead. A medida que o relacionamento vai crescendo, é preciso dar cada vez mais atenção ao lead. Com uma relação já amadurecida, a equipe de marketing passa as informações para a equipe

de vendas que deverá decidir em quais leads deve investir tempo com uma grande chance de conversão em clientes. Por fim, mantendo uma boa relação com a equipe de vendas, o lead percebe o quão relevante é o produto/serviço ofertado e fecha negócio. Existe ainda a etapa do pós-venda que é responsável por fidelizar o cliente a marca. Esse cuidado pode fazer com que o cliente se torne um divulgador da marca por fazê-lo sentir-se especial e único, apresentando o produto/serviço a outras pessoas.

A passagem de leads quentes da equipe de marketing para a equipe de vendas deve ser feita de maneira rápida e eficiente, o que é um grande desafio. Essa etapa é indispensável para o *Inbound* marketing e pode ser auxiliada através da técnica de Lead Scoring: trata-se de uma pontuação que os Leads recebem para identificar os que estão mais propensos a compra e que geralmente é resultado da relação entre as informações de perfil do lead e interesse. Isso ajuda bastante a equipe de marketing, pois será muito mais rápido analisar os Leads e entregar as melhores oportunidades aos vendedores. A nota atribuída no lead scoring pode ser obtida de várias maneiras, desde a análise manual onde a equipe de marketing irá usar sua experiência e intuição para determinar os leads quentes um a um. Existe também a maneira automática preditiva, cujo software é quem analisará as informações da base de dados definindo critérios e pesos automaticamente. Ainda assim, é necessário ajustar e informar ao software quais critérios serão mais relevantes: é preciso analisar minuciosamente pois é nessa etapa que pode ser comprometida a eficácia do lead scoring (RESULTADOS DIGITAIS, 2015).

Este trabalho propõe uma seleção automática preditiva dos leads quentes usando a base de dados onde foram colhidas as informações dos leads e também de clientes, excluindo a informação de lead scoring. Através do uso de técnicas de aprendizado de máquina supervisionado, que vão usar como critério chave os eventos que foram salvos na base de dados a cada troca de informações entre a empresa e o lead, a quantidade de vezes em que eles foram realizados e a informação se aquele indivíduo é lead ou cliente. Esta etapa será detalhada na Seção 3.

2.3 Reconhecimento de padrões

Dados são fatos que foram coletados e armazenados, formando assim uma informação com algum significado. Um conjunto com vários dados pode oferecer informações importantes de diversas maneiras. Para que essas informações tenham algum sentido, é necessário saber interpretá-las e, um ponto crucial para interpretação desses dados é observar se existem padrões entre eles.

Os padrões se apresentam de diversos modos; pode ser um som, gosto, forma, tamanho, entre outros que podem facilmente ser identificados pelo ser humano. É possível então, reconhecer, naturalmente, a chuva pela sua forma e som assim como, reconhecer prédios pela sua forma e tamanho. Essa capacidade cognitiva pode ser reproduzida artificialmente através de cálculos estatísticos combinados em algoritmos de computador, que ajudam a reconhecer padrões diversos em grandes quantidades de dados. O Reconhecimento de Padrões é um campo da ciência que tem como objetivo, apresentar uma representação mais simples de um conjunto de dados, através de suas características mais relevantes, possibilitando a partição em classes (DUDA et al., 2001). Ou seja, é possível classificar um objeto entre categorias e classes, observando suas características (THEODORIDIS; KOUTROUMBAS, 2003), através de registros em uma base de dados.

A classificação correta pode ajudar na tomada de decisão ao qual está relacionada, como por exemplo, o risco que um banco corre ao liberar crédito para um determinado cliente. O banco pode verificar se trata-se de um bom ou mau pagador para decidir se libera o crédito. Porém, tratando-se de um cliente novo, o banco não teria acesso a essa informação, portanto seria necessário levar em consideração outras características para calcular o risco.

As técnicas e classificação podem ser de dois tipos: supervisionadas, no qual se faz necessário obter previamente, informações de um especialista com grande conhecimento do que está sendo estudado; e as não supervisionadas, que podem abdicar desse conhecimento prévio (REBOUÇAS, 2011). Neste trabalho, faremos uso apenas das técnicas de classificação supervisionadas, tendo em vista que as classes foram definidas através de uma base de dados formada por exemplos dos padrões conhecidos.

2.3.1 Classificação supervisionada

Na classificação supervisionada, uma classe é atribuída a uma instância desconhecida que deve ser similar a instâncias conhecidas e previamente rotuladas, o qual chama-se conjunto de treinamento. Ou seja, são fornecidos exemplos ao algoritmo de aprendizado onde E = {E_1, E_2,...En}, e cada exemplo Ei ∈ E, possui um rótulo associado. O objetivo da máquina de aprendizado é induzir um classificador que separe os padrões de acordo com a classe que lhes foi rotulada.

A fim de facilitar a compreensão, pode-se recorrer ao seguinte exemplo adaptado de (RUSS, 1990). Trata-se de um sistema baseado em análise de imagens para classificar frutas, no qual (ameixas, maçãs, limões, melões) são as classes. As características analisadas foram o "tamanho" e a "cor" de cada fruta, que por sua vez, foram escolhidas de acordo com a classificação, levando em consideração a observação prévia de frutas já conhecidas e suas respectivas classes (conjunto de treinamento). De acordo com a Figura 2, observando a característica "tamanho", foi possível identificar maçãs e melões. Entretanto, ameixas e limões são frutas de tamanhos próximos e podem ser confundidas. Com o uso da característica "cor" foi possível determinar as ameixas e os limões.

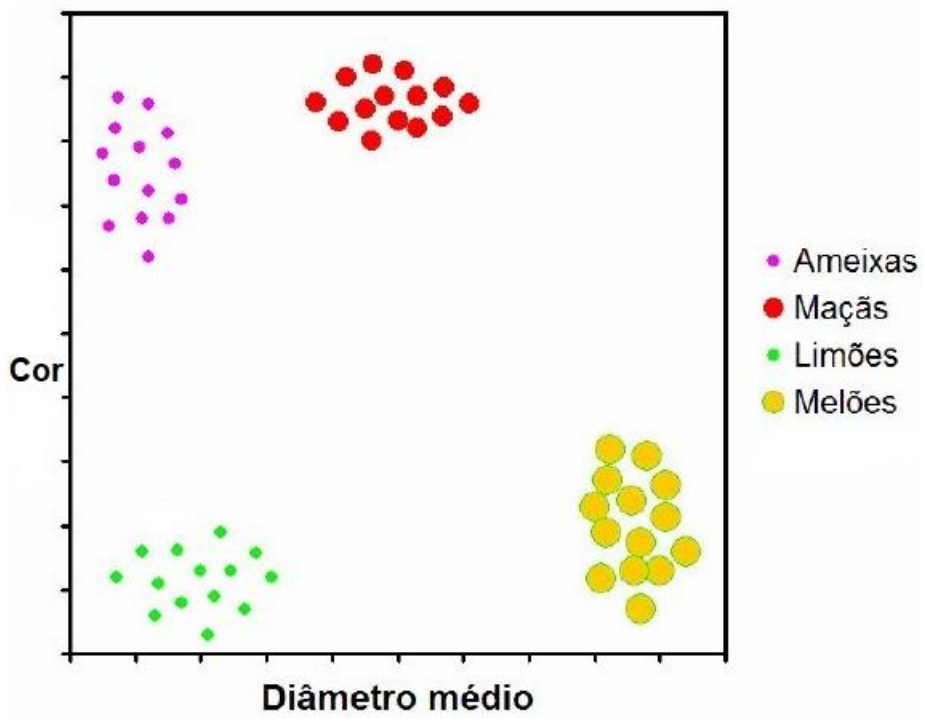

Figura 2 - Exemplo de classificação supervisionada.

No exemplo em questão, o algoritmo aprende as características das frutas rotuladas, padroniza medidas de similaridade de cada fruta, que representa uma classe, e por fim, prevê uma fruta não rotulada, ou seja, classifica uma fruta de acordo com suas características.

Desta forma, dispondo de classes rotuladas que compõem um conjunto de treinamento e de um conjunto de características a um classificador, a identificação de um objeto desconhecido pode ser feita automaticamente através de um procedimento de classificação supervisionada (DUDA et al., 2001) (GONZALEZ; WOODS, 2002).

Tendo em vista a limitação de tempo para a realização dos experimentos deste trabalho, foram escolhidos dois algoritmos para fazer a classificação supervisionada: Árvore de decisão, no qual é possível fazer a seleção de modelos para determinar qual consegue classificar melhor as instâncias, levando em consideração o ganho de informação de cada atributo (será detalhado na próxima subseção), e o algoritmo Naive Bayes, no qual os atributos não são correlacionados durante o processo de classificação.

2.3.2 Árvore de decisão

Uma decisão deve ser tomada quando se tem um problema que possui mais de uma solução (GOMES et al., 2002). De modo geral, em computação, Árvore de decisão é uma técnica de aprendizado de máquina no qual um problema complexo é decomposto em subproblemas mais simples e, recursivamente, essa estratégia é aplicada em cada subproblema (GAMA, 2000). Para GOEBEL e GRUENWALD (1999), árvores de decisão são estruturas de dados formadas por um conjunto de elementos que armazenam informações chamadas nós e cada nó interno (não terminal), representa uma decisão sobre o item de dado considerado. Com isso, uma Árvore de decisão pode ser formada por um conjunto de regras que serão induzidas de acordo com as condições determinadas, optando sempre por árvores menores. Árvores grandes podem implicar em problemas de sobre-ajuste, por isso a importância de otimizar os parâmetros que formam o conjunto de regras (STEINER et al., 2004). Neste trabalho, o ajuste de parâmetros será realizado no Capítulo 3.

Para elucidar melhor, adotou-se a exemplificação do funcionamento do clássico algoritmo ID3 (QUINLAN, 1979). As árvores são construídas de cima para baixo, começando pelo atributo raiz, que será o que tiver o melhor ganho de informação.

Para isso são feitos cálculos de entropia, que em aprendizado de máquina, mede a impureza de uma coleção arbitrária de exemplos. As fórmulas a seguir foram extraídas de MICTHELL (1997).

Dado uma coleção S contendo exemplos positivos e negativos de algum conceito alvo, a entropia S relativa a esta classificação booleana é:

$$Entropia(S) \equiv -p_+ \log_2 p_+ - p_- log_2 p_-$$

Onde p_+ é a proporção de exemplos positivos em *S* e p_- é a proporção de exemplos negativos em *S*. Para multiclasses, pode-se generalizar a fórmula da entropia como sendo:

$$Entropia(S) \equiv \sum_{i=0}^{n} -p_i \log_2 p_i$$

Onde i representa cada classe. O ganho de informação é a redução esperada na entropia devido a ordenação sobre um atributo A, ou seja, é a redução esperada na entropia causada pela *partição dos exemplos de acordo com este atributo* A representada pelo conjunto S_v.

$$Ganho(S, A) \equiv Entropia(S) - \sum_{v \in Valores\ (A)} \frac{|S_v|}{S} Entropia(S_v)$$

Para encontrar a raiz da árvore, calcula-se o ganho de informação para todos os atributos da base de dados. O atributo que tiver o maior ganho de informação será o atributo raiz da árvore e os valores deste atributo serão os ramos da árvore. Para cada ramo é selecionado todo subconjunto dele. Com base nesse subconjunto, calcula-se novamente o ganho, dessa vez para cada elemento do subconjunto e assim sucessivamente até chegar no atributo que rotula cada instância, ou seja, o que determina se ela é positiva ou negativa, no caso de uma classificação booleana, ou o que estabelece a qual classe pertence numa classificação de multiclasses. Em outras palavras, recursivamente, repete-se o processo de selecionar um novo atributo e particionar os exemplos de treinamento para cada nó descendente não terminal, até encontrar um nó terminal. São utilizados somente os exemplos de treinamento associados com este nó. Todos os atributos devem ser incluídos ao longo do caminho da árvore e a medida em que vão sendo incorporados, são também excluídos, pois

só devem aparecer uma vez ao longo de qualquer caminho da árvore. Por fim, o processo é repetido para cada ramo da árvore até que os exemplos de treinamento associados com este ramo, tenham todos o mesmo valor do atributo chave.

MITCHELL (1997) aponta que o algoritmo ID3 é descrito como um método de procura dentro em um espaço de hipóteses, onde a hipótese se ajusta aos exemplos de treinamento. Ou seja, o método encontra uma possibilidade dentro do conjunto de possibilidades de árvore. O espaço de hipóteses buscado pelo ID3 é o conjunto de Árvores de decisão possíveis. A seguir é apresentado o pseudocódigo do algoritmo ID3, traduzido de MITCHELL (1997):

Início

Crie um nodo ***Raiz*** para a árvore

Se todos os ***Exemplos*** são positivos

Então retorna a ***Raiz*** da árvore com o rótulo = **sim**

Se todos os ***Exemplos*** são negativos

Então retorna a **Raiz** da árvore com o rótulo = ***não***

Se ***Atributos*** for vazio

Então retorna a ***Raiz*** da árvore com o rótulo = valor mais comum do ***Atributo-objetivo*** em ***Exemplos***

Senão

A ← um atributo de **Atributos** que melhor classifica **Exemplos** (atributo de decisão)

Raiz ← **A** (rótulo = atributo de decisão **A**)

Para cada possível valor v_i de **A** faça

Acrescenta um novo arco abaixo da ***Raiz***, correspondendo à resposta ***A*** = v_i

Seja ***Exemplos*** $_{vi}$ o subconjunto de ***Exemplos*** que têm valor v_i para A

Se ***Exemplos*** $_{vi}$ for vazio

Então acrescenta na extremidade do arco um nodo folha com rótulo = valor mais comum do ***Atributo-objetivo*** em ***Exemplos***

Senão acrescenta na extremidade do arco a sub árvore

ID3(***Exemplos*** $_{vi}$, **Atributo-objetivo**, **Atributos - {A})**

Retorna **Raiz** (aponta para a árvore)

Fim

Pensando em ilustrar o algoritmo ID3, adotou-se um exemplo adaptado de MITCHELL (1997), que definirá se uma pessoa irá jogar tênis ou não. A base possui cinco colunas, onde quatro delas contém as características (*features*) que serão usadas para construção da árvore (Aspecto, Temperatura, Umidade e Vento) e a

coluna contendo o atributo alvo, ou seja, o atributo booleano com o rótulo se a pessoa irá jogar ou não. No conjunto de treinamento, tem-se 14 exemplos, dos quais 9 contém o rótulo “Sim” e 5 contém o rótulo “Não”.

Aspecto	Temp.	Umidade	Vento	Jogar Tênis
Sol	Quente	Elevada	Fraco	Não
Sol	Quente	Elevada	Forte	Não
Nuvens	Quente	Elevada	Fraco	Sim
Chuva	Ameno	Elevada	Fraco	Sim
Chuva	Fresco	Normal	Fraco	Sim
Chuva	Fresco	Normal	Forte	Não
Nuvens	Fresco	Normal	Fraco	Sim
Sol	Ameno	Elevada	Fraco	Não
Sol	Fresco	Normal	Fraco	Sim
Chuva	Ameno	Normal	Forte	Sim
Sol	Ameno	Normal	Forte	Sim
Nuvens	Ameno	Elevada	Forte	Sim
Nuvens	Quente	Normal	Fraco	Sim
Chuva	Ameno	Elevada	Forte	Não

Figura 3 - Conjunto de treinamento para aplicação do algoritmo ID3.

Primeiramente, calcula-se a entropia (impureza) do conjunto de treinamento, como segue:

$$Entropia([9_+, 5_-]) = -\left(\frac{9}{14}\, log_2 \frac{9}{14} - \frac{5}{14} log_2\, \frac{5}{14}\right)$$

$$Entropia([9_+, 5_-]) = 0.940$$

Deve-se calcular ainda, a entropia para cada atributo de acordo com seus elementos, levando em consideração o atributo alvo. No caso do atributo Vento, têm-se dois elementos, “Forte” $[3_+, 3_-]$, onde aparecem seis ocorrências no qual em três delas o indivíduo jogou tênis e nas outras 3 vezes não jogou e “Fraco” $[6_+, 2_-]$, com mais oito ocorrências, das quais em seis delas a pessoa jogou tênis e nas outras duas não jogou, completando os 14 elementos do conjunto de treinamento. Aplicando a fórmula da entropia, tem-se $Entropia(S_{Forte}) = 0.811$ e $Entropia(S_{Fraco}) = 1$. Agora, deve-se calcular o ganho de informação do atributo Vento:

$$Ganho(S, vento) = Entropia(S) \ - \frac{8}{14} \ Entropia(S_{Fraco}) - \frac{6}{14} \ Entropia(S_{Forte})$$

$$Ganho(S, vento) = 0.940 - \frac{8}{14} * 0.811 - \frac{6}{14} * 1$$

$$Ganho(S, vento) = 0.048$$

Desse modo calcula-se o ganho de informação para todos os atributos do conjunto de treinamento:

$$Ganho(S, aspecto) = \ 0.246$$

$$Ganho(S, temperatura) \ = \ 0.029$$

$$Ganho(S, umidade) \ = \ 0.151$$

$$Ganho(S, vento) \ = \ 0.048$$

Observando os resultados, foi identificado que o maior ganho de informação foi do atributo Aspecto, logo, esse será o atributo raiz da árvore. Após isso, o algoritmo deve observar os valores do atributo Aspecto (Sol, Nuvens, Chuva). Cada um desses valores é um ramo da árvore e para cada um deles deve ser calculado o ganho de informação em relação aos demais atributos (Temperatura, Umidade e Vento). Na primeira ocorrência de Sol, o valor do atributo Temperatura foi “Quente”, o valor para Umidade foi “Elevada”, o valor para Vento foi “Fraco”. Ou seja, para cada ocorrência do valor “Sol” do atributo Aspecto, deve-se observar os valores dos demais atributos em função do atributo alvo, sendo representado por $[2_+, 3_-]$

Aspecto	Temp.	Umidade	Vento	Jogar Tênis
Sol	Quente	Elevada	Fraco	Não
Sol	Quente	Elevada	Forte	Não
Sol	Ameno	Elevada	Fraco	Não
Sol	Fresco	Normal	Fraco	Sim
Sol	Ameno	Normal	Forte	Sim

Figura 4 - Recorte com as ocorrências do valor “Sol” para o atributo Aspecto.

Para saber qual atributo ocupará o próximo ramo, deve-se calcular novamente o ganho de informação para cada valor do atributo Aspecto em relação aos demais atributos.

$$Ganho(S_{Sol}, Umidade) = 0.970 - \left(\frac{3}{5}\right) * 0 - \left(\frac{2}{5}\right) * 0 = 0.970$$

$$Ganho(S_{Sol}, Temperatura) = 0.970 - \left(\frac{2}{5}\right) * 0 - \left(\frac{2}{5}\right) * 1 - \left(\frac{1}{5}\right) * 0 = 0.570$$

$$Ganho(S_{Sol}, Vento) = 0.970 - \left(\frac{2}{5}\right) * 1 - \left(\frac{3}{5}\right) * 0.918 = 0.019$$

Analisando os resultados, observou-se que o atributo com maior ganho em relação ao ramo Sol, foi a Umidade. Repete-se o processo recursivamente sem repetir os atributos que já estão na árvore, até chegar no atributo alvo e obter o rótulo Sim ou Não. A partir dessa sequência de cálculos, a árvore pode ser representada da seguinte forma:

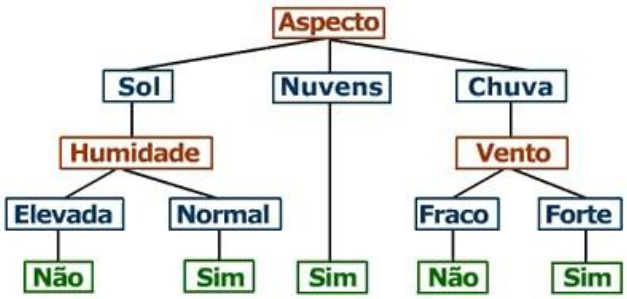

Figura 5 - Árvore de decisão para o exemplo do conjunto de treinamento da Figura 3.

A biblioteca SCIKIT-LEARN (2019), utiliza na implementação da Árvore de decisão uma versão otimizada do algoritmo Árvores de classificação e Regressão (*Classification and Regression Trees*)(CART), proposto por BREIMAN et al. (1984). Trata-se de um algoritmo semelhante ao C4.5 (QUINLAN, 1993), sucessor do algoritmo ID3 detalhado neste trabalho. O CART consiste em uma técnica para induzir árvores de classificação e regressão. Sua principal vantagem é a capacidade de relacionar os dados, mesmo que aparentemente não exista relação entre eles, assim como a capacidade de apresentar como resultado a própria árvore de decisão, facilitando a compreensão de seu funcionamento (FONSECA, 1994).

A abordagem utilizada no algoritmo CART difere da encontrada em outros algoritmos, que utilizam a pré-poda. O CART constrói árvores binárias, com questões que contém como possíveis respostas apenas "sim" ou "não", desde o nó raiz até os

nós-folhas. Deste modo, a árvore é expandida de acordo com as regras pré-definidas (hiperparâmetros do modelo), realizando a pós-poda em função da diminuição do custo-complexidade. Segundo BREIMAN et al. (1984), esta técnica apresenta um ótimo desempenho, construindo árvores de fácil entendimento e bastante precisas, contendo uma ótima capacidade de generalização.

2.3.3 Naive Bayesian Learning

Dentre os algoritmos de classificação supervisionada, Naive Bayes é provavelmente o classificador mais utilizado. Sua tradução sugere tratar-se de um algoritmo "ingênuo" (*naive*) que usa o teorema de Thomas Bayes, ministro inglês do século XVIII, através da aplicação de fórmulas estatísticas e cálculos de probabilidades para que se possa fazer uma classificação (MITCHELL, 1997).

A premissa "ingênua" parte do motivo que o classificador desconsidera completamente a correlação entre os atributos (*features*). Por exemplo, se determinado animal é considerado um "Gato", se ele for "pequeno", "peludo" e "miar", o algoritmo não levará em consideração a correlação entre esses fatores, tratando-os de forma independente (CANDIAGO, 2017). A princípio, pode-se parecer simplista, mas este classificador é capaz de reportar o melhor desempenho em várias tarefas de classificação.

Para entender o funcionamento do algoritmo, alguns conceitos devem ser explicados. As fórmulas a seguir foram extraídas de MITCHELL (1997). O primeiro conceito é o de probabilidade condicional, que é a probabilidade que um evento aconteça dado que outro evento aconteceu.

$$P(A|B)$$

A fórmula representa a probabilidade de A acontecer, dado que B ocorreu, ou seja, a probabilidade da interseção entre os eventos B e A dividido pela probabilidade de B ocorrer e multiplicado pela probabilidade da interseção entre B e A.

$$P(A|B) = \frac{P(B \cap A\ )}{P(B)} P(B \cap A) = P(A|B) * P(B)$$

Isto é igual a probabilidade de A acontecer dado que B ocorreu multiplicado pela probabilidade de B acontecer. Aplicando a propriedade comutativa, tem-se que:

$$P(B \cap A) = P(A \cap B)$$

Com isso,

$$P(A|B) * P(B) = P(B|A) * P(A)$$

Simplificando, chega-se ao Teorema de Bayes:

$$P(A|B) = \frac{P(B \mid A) * P(A)}{P(B)}$$

Com intuito de elucidar melhor, será feita a aplicação do teorema, com o mesmo exemplo da seção anterior, adaptado de MITCHELL (1997), onde utilizou-se o conjunto de treinamento para classificar se uma pessoa jogaria tênis. Sabendo que o classificador bayesiano não correlaciona os atributos, qual a probabilidade de ocorrer o evento alvo "Sim" (9 ocorrências das 14 instâncias), ou seja, o indivíduo jogar tênis, dado que Aspecto = "Sol" (duas ocorrências em jogar = "Sim", $2/9$); Temperatura = "Fresco", ou seja, 3 ocorrências ($3/9$); Umidade = "Elevada" que aparece 3 vezes ($3/9$) e Vento = "Forte" que acontece 3 vezes ($3/9$)? Aplicando o Teorema de Bayes, tem-se que:

$$\frac{P(Sol \mid Sim) * P(Fresco \mid Sim) * P(Elevada \mid Sim) * P(Forte \mid Sim) * P(Sim)}{P(Sol) * P(Fresco) * P(Elevada) * P(Forte)}$$

Onde P(Sol) = $5/14$, visto que houveram 5 ocorrências do Aspecto "Sol" na base de dados toda. Disso, conclui-se que P(Fresco) = $4/14$, P(Elevada) = $7/14$ e P(Forte) = $6/14$. Todos os valores foram extraídos da Figura 3. Portanto:

$$\frac{2/9 * 3/9 * 3/9 * 3/9 * 9/14}{5/14 * 4/14 * 7/14 * 6/14} = \frac{0.0053}{0.02186} = 0.242$$

Baseado nesse resultado, o classificador retornou 24% de chance do indivíduo jogar tênis. E qual seria a probabilidade de não jogar tênis (5 ocorrências dentre as 14 instâncias) dados os mesmos valores para cada atributo?

$$\frac{P(Sol \mid Não) * P(Fresco \mid Não) * P(Elevada \mid Não) * P(Forte \mid Não) * P(Não)}{P(Sol) * P(Fresco) * P(Elevada) * P(Forte)}$$

Extraindo os valores da Figura 3, tem-se que:

$$\frac{3/5 * 1/5 * 4/5 * 3/5 * 5/14}{5/14 * 4/14 * 7/14 * 6/14} = \frac{0.0206}{0.02186} = 0.940$$

Portanto, o índice de 94% é maior do que o complemento da probabilidade de jogar, pois ele é calculado com base nos valores dos atributos da instância que foi classificada. Com isso, conclui-se que a probabilidade do atributo alvo jogar tênis ser “Não” é maior do que a probabilidade de ocorrer o valor “Sim” para este atributo.

2.3.4 Classificação em uma base de dados desbalanceada

Uma problemática recorrente ao utilizar algoritmos de classificação pode estar no desbalanceamento dos dados. Uma base é tida como desbalanceada, quando existem muitas instâncias de uma certa classe em relação às da outra (1 instância da classe positiva para 100 da negativa, 1 para 1000, ou mais) (CHAWLA et al., 2002). Para obter melhores resultados de classificação, é importante que a base de dados esteja balanceada para que o algoritmo possa aprender com uma quantidade de instâncias negativas e positivas equivalentes, e, assim, possa classificar corretamente as novas instâncias. Portanto, um modelo de classificação é eficaz quando consegue identificar corretamente instâncias que pertencem a uma determinada categoria.

Para compreender melhor a problemática da classificação em dados desbalanceados, foi adotado o exemplo de um conjunto de dados de imagens de mamografia. Ao analisar as imagens observou-se duas classes: a classe minoritária, chamada de positiva, que representava 2% das instâncias analisadas e na qual faziam parte as pessoas com câncer de mama; e a classe majoritária, chamada de negativa, e que representava 98% das instâncias e continha as pessoas saudáveis, ou seja, as que não possuíam câncer de mama. Um bom classificador deve obter a taxa de acerto mais próxima de 100% tanto para a classe minoritária, quanto para a majoritária. Entretanto, é comum ver classificadores obtendo uma taxa de acerto próxima dos 100% para a classe majoritária e entre 0 a 10% para a classe minoritária (HE; GARCIA, 2009). Com isso, as instâncias positivas são classificadas erroneamente como negativas, ou seja, pessoas com câncer serão identificadas como pessoas saudáveis. Isso ocasionaria transtornos gravíssimos e possivelmente irreversíveis a vida dessas pessoas que deveriam ser submetidas a tratamento imediato.

A partir do exemplo, pode-se identificar dois possíveis erros: a classificação de uma instância positiva como negativa e a identificação de uma instância da classe

negativa como positiva. Com isso, o desempenho de sistemas de classificação em bases de dados com classes desbalanceadas tende a ser baixo. Existem, porém, algoritmos que tratam a problemática do desbalanceamento de dados fazendo uma sobre-amostragem da classe minoritária ou uma subamostragem da classe majoritária.

Durante a realização dos experimentos deste trabalho, foi necessário fazer o balanceamento da base de dados aplicando o algoritmo de Técnica de Sobre-amostragem Sintética de Minoria (*Synthetic Minority Over-sampling Technique*)(SMOTE), (CHAWLA et al., 2002). Trata-se de um algoritmo que cria elementos sintéticos da classe minoritária, através de exemplos já rotulados, para controlar a criação das amostras artificiais com objetivo de balancear a base de dados e minimizar possíveis efeitos de sobre-ajuste (HE; GARCIA, 2009).

Analisando o algoritmo tem-se que $|S| = |S_{pos}| + |S_{neg}|$ no qual S equivale a um conjunto de treinamento com n instâncias, assim, $|S| = n$, S_{pos} e S_{neg} correspondem aos elementos das classes positiva e negativa, respectivamente. Para o subconjunto S_{pos}, que representa a classe minoritária, são considerados os k-vizinhos mais próximos em cada instância $x_i \in S_{pos}$ para algum valor inteiro de k. Com intuito de facilitar a compreensão, recorreu-se ao exemplo da Figura 6. O ponto vermelho representa uma instância $x_i \in S_{pos}$. A cada iteração, uma instância x_i da classe minoritária é escolhida aleatoriamente. Os pontos azuis representam os k-vizinhos de x_i. Primeiramente é calculada a distância euclidiana entre a instância x_i em quentão e os k-vizinhos, com a finalidade de encontrar os vizinhos mais próximos. Em seguida, essa distância é multiplicada por um valor aleatório entre 0 e 1. O vetor gerado é aplicado a instância x_i, onde é selecionado um ponto entre x_i e o k-vizinho mais próximo. Por fim, a amostra sintética é criada. Na Figura 6, são representadas pelos pontos verdes. O processo é repetido até que o balanceamento esteja concluído.

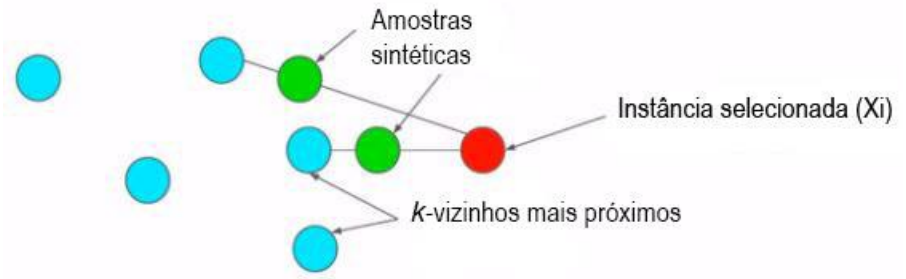

Figura 6 - Exemplo de sobre-amostragem sintética com SMOTE.

2.3.5 Seleção de atributos

As técnicas de seleção de atributos (SA) são estudadas desde meados dos anos 70, nas áreas de reconhecimento de padrões, aprendizado de máquina e mineração de dados (LIU, YU, 2002). Na classificação supervisionada os atributos de uma base de dados devem formar um modelo que consiga predizer instâncias não rotuladas de forma correta (PIRAMUTHU, 2006).

É comum encontrar bases de dados contendo uma grande quantidade de atributos. Essa alta dimensionalidade (grande quantidade de características para cada instância), prejudica a predição dos classificadores de aprendizado supervisionado. Ou seja, a classificação é dificultada ao tentar encontrar padrões num conjunto de dados que contém uma grande quantidade de atributos.

RODRIGUES (2016) afirma que para um dado tamanho de amostra, deve-se existir um número máximo de atributos. Ao ultrapassar esta quantidade máxima, o desempenho do classificador é degradado, ao invés de ser melhorado. Com isso, o desempenho de um classificador tende a diminuir em função de uma grande quantidade de características, conforme ilustrado na Figura 7.

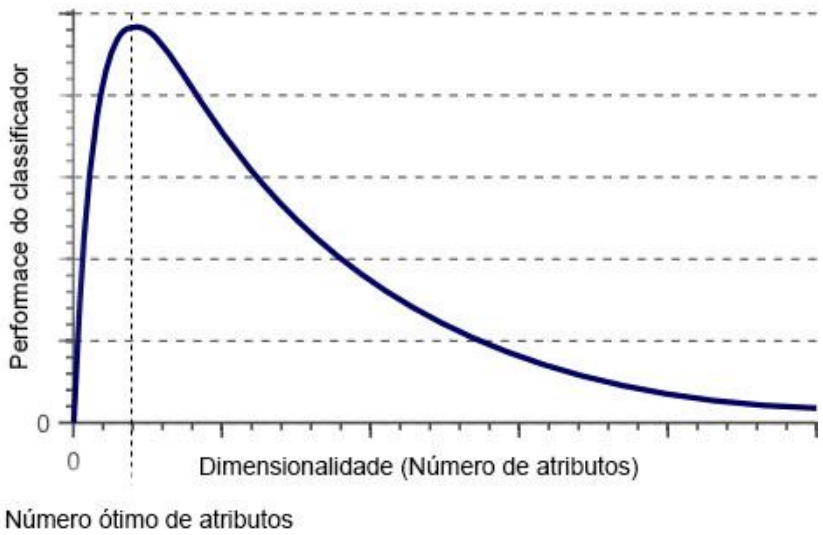

Figura 7 - Curva de desempenho do classificador em função da dimensionalidade.
Traduzido de LENINE (2017)

O objetivo da seleção de atributos é encontrar as características (*features*) mais relevantes para a classificação, tendo assim, uma importante função na etapa de pré-processamento. Através desta técnica é possível ordenar os atributos a partir de

algum critério de importância, assim como reduzir o problema da alta dimensionalidade do espaço onde eles serão procurados.

A seleção de atributos pode ser realizada pelo método de visualização dos dados por um especialista da área ou através de métodos simples de estatística, como medidas de médias e desvios-padrões (MICHALSKI et al., 1998). Como resultado da realização da seleção de atributos, descobriu-se que, uma vez selecionados corretamente, a qualidade dos dados pode ser melhorada e em consequência disso, a performance dos classificadores pode ser aperfeiçoada (WITTEN et al., 2016).

Para a realização deste trabalho, será utilizado o método de visualização dos dados realizado por um especialista. Sua descrição detalhada será apresentada no Capítulo 3, Experimentos.

2.3.6 Seleção de modelos

Um modelo pode ter muitos hiperparâmetros e encontrar a melhor combinação entre eles deve ser tratado como um problema de pesquisa. Os hiperparâmetros são variáveis utilizadas para controlar o treinamento de um determinado classificador, por exemplo, o número de amostras necessárias para dividir um ramo em uma árvore de decisão. Embora existam muitos algoritmos de otimização/ajuste de hiperparâmetros este trabalho apresentará duas soluções disponíveis na biblioteca de aprendizado de máquina Scikit-learn para a linguagem de programação Python. A melhor solução será escolhida.

Comumente, a pesquisa em grade (*grid search*) é vista como a técnica de busca mais utilizada, por tratar-se de um método que tenta exaustivamente todas as combinações de hiperparâmetros pré-especificados e retorna a melhor opção, ou seja, a que melhor classifica o conjunto de teste. Entretanto, testar exaustivamente várias combinações tende a ser muito dispendioso, em termos computacionais. Para que a pesquisa em grade seja viável, muitas vezes é necessário restringir o número de combinações, limitando muito o quanto pode-se explorar dos hiperparâmetros.

Outro método de busca pelos melhores hiperparâmetros bastante utilizado é a Pesquisa aleatória (*Random search*). Originalmente, o método de pesquisa aleatória foi desenvolvido por BERGSTRA e BENGIO (2012). Ao contrário da pesquisa em grade, no qual valores fixos de hiperparâmetros são experimentados combinatoriamente, na pesquisa aleatória pode-se atribuir, para cada hiperparâmetro,

uma distribuição de valores mais ampla dentre algum intervalo. Na pesquisa aleatória, pode-se pré-especificar o "orçamento computacional" através do parâmetro *n_iter*, que controla quantas configurações de parâmetros diferentes serão testadas.

Para elucidar melhor, recorreu-se ao exemplo apresentado na Figura 8, adaptado de LATYSHEVA e RAVARANI (2016):

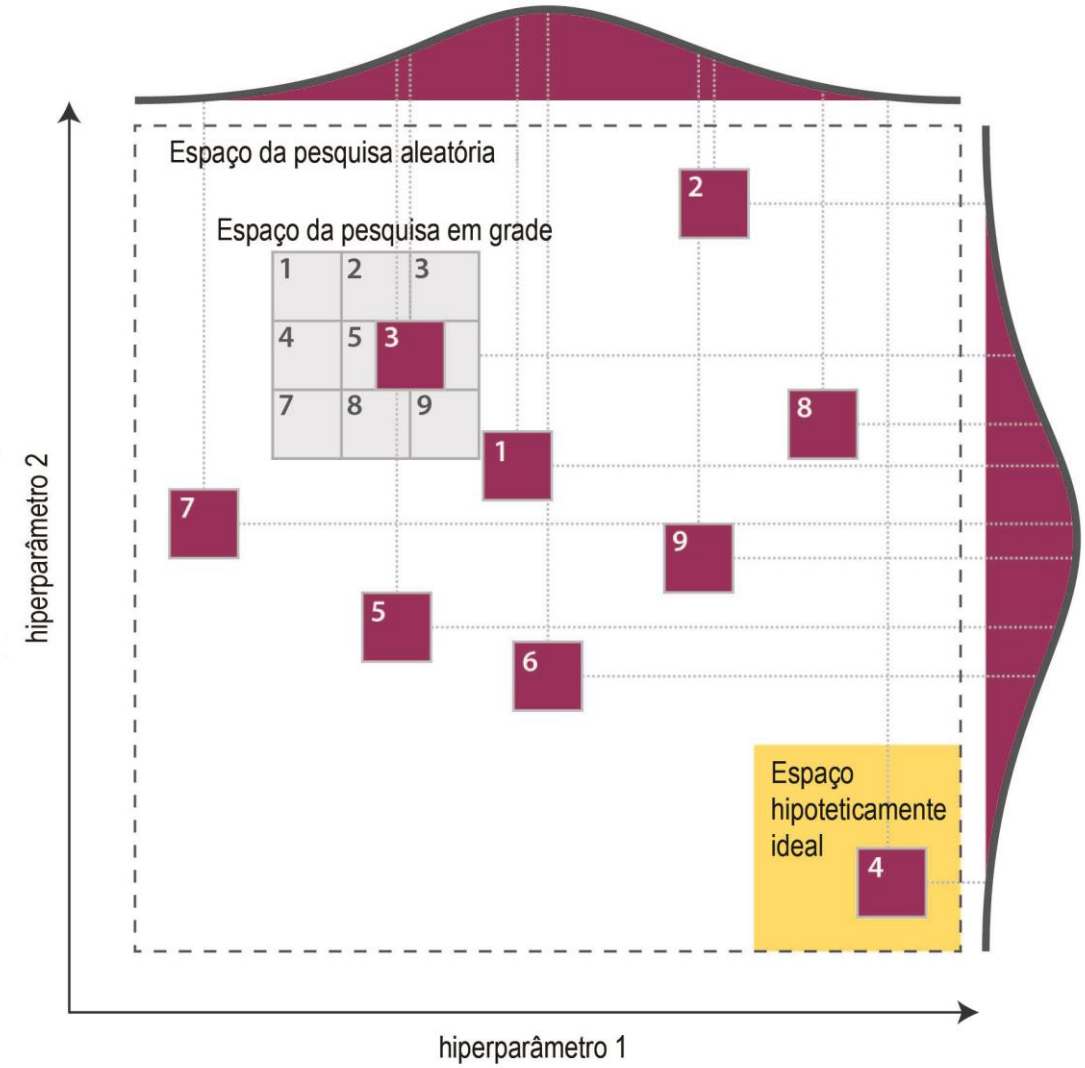

Figura 8 - Estratégias de pesquisa em grade e pesquisa aleatória.
Traduzido de LATYSHEVA e RAVARANI (2016)

A imagem ilustra o espaço amostral dos métodos de pesquisa em grade e pesquisa aleatória, com orçamento computacional limitado a nove iterações para cada abordagem. Em outras palavras, foram feitas nove buscas pelo conjunto de hiperparâmetros que melhor classifica as instâncias não rotuladas. A pesquisa aleatória testa os valores de hiperparâmetros em um espaço amostral bem mais

amplo, visto que neste método é definido um conjunto de possibilidades para cada hiperparâmetro com um valor inicial e um valor final. A cada iteração é feita uma seleção aleatória de um valor dentre esse conjunto de possibilidades. O processo é repetido nove vezes, obtendo assim nove combinações diferentes de valores dos hiperparâmetros 1 e 2. Neste método, pode-se observar que toda distribuição foi amostrada, o que significa que teve-se a chance de experimentar regiões mais distantes onde atingiu-se a combinação ideal entre os hiperparâmetros.

A pesquisa em grade testou apenas 3 valores para cada hiperparâmetro. Isso porque seu conjunto de possibilidades contém valores fixos e o valor de cada hiperparâmetro é testado de acordo com a disponibilidade do orçamento computacional. Sabe-se que esses valores não precisam ser próximos uns dos outros e, de fato, o melhor modelo, poderia ser alcançado. Contudo, a pesquisa em grade limita bastante os valores dos hiperparâmetros que podem ser explorados. No exemplo em questão, a pesquisa aleatória alcançou um espaço dezesseis vezes maior. BERGSTRA e BENGIO (2012) afirmam que, se uma mesma configuração for testada para diferentes conjuntos de dados, os espaços importantes também serão diferentes, logo é mais fácil chegar ao melhor modelo com a pesquisa aleatória.

2.3.7 Validação cruzada aninhada

Uma das etapas mais importantes em um projeto de aprendizado de máquina é a etapa de validação dos resultados. Usualmente, no Reconhecimento de padrões, a validação de dados é usada para otimizar ou selecionar os melhores parâmetros de um algoritmo (WITTEN et al., 2005). É comum que os resultados sejam avaliados antes que o algoritmo classifique instâncias desconhecidas. Essa medição de desempenho é feita na etapa de validação, comumente chamada etapa de teste. Portanto, o algoritmo de classificação deve possuir três etapas: treinamento, validação e classificação.

CAWLEY e TALBOT (2010), apontam em seu artigo que quando um conjunto de teste é definido para selecionar os parâmetros e avaliar o modelo, corre-se o risco de obter resultados super otimistas. Por esse motivo, se um conjunto de testes for usado para selecionar os parâmetros de um modelo, será preciso um conjunto de testes diferente para obter uma avaliação imparcial do modelo selecionado.

Ocorre ainda que, em muitos casos, o fornecimento dos dados que formam os conjuntos de treinamento e teste é bastante limitado. Isso faz com que o conjunto de testes possa não ser um bom indicador de desempenho. Quando se divide uma base de dados nesses dois conjuntos, é interessante usar o máximo de dados disponíveis (BISHOP, 2006).

A solução pode ser encontrada fazendo uso da técnica de validação cruzada aninhada. Também conhecida como “CV aninhado” (*Nested Cross Validation*), é comumente usada para treinar um modelo onde os hiperparâmetros precisam ser otimizados. A validação cruzada aninhada estima o erro de generalização do modelo subjacente e a pesquisa pelos hiperparâmetros. A escolha dos parâmetros feita em uma validação cruzada não aninhada, polariza o modelo, visto que a base de dados “vaza” informações do conjunto de testes, gerando uma pontuação otimista (SCIKIT-LEARN, 2019).

A validação cruzada aninhada consiste em dois loops de validação cruzada, sendo um *k-fold* interno e outro externo. Cada *k-fold* é subdividido em dobras que serão utilizadas uma vez como validação. São feitas 10 validações cruzadas com o loop externo e para cada iteração do CV externo, são feitas validações cruzadas com o *k-fold* interno, repetindo de acordo com a quantidade de dobras setada. O *k-fold* interno, é responsável pela seleção de modelos/ajuste dos hiperparâmetros (semelhante ao conjunto de validação), ou seja, é o conjunto de treinamento. Já o *k-fold* externo é utilizado para avaliar o modelo selecionado pela validação cruzada interna (conjunto de teste).

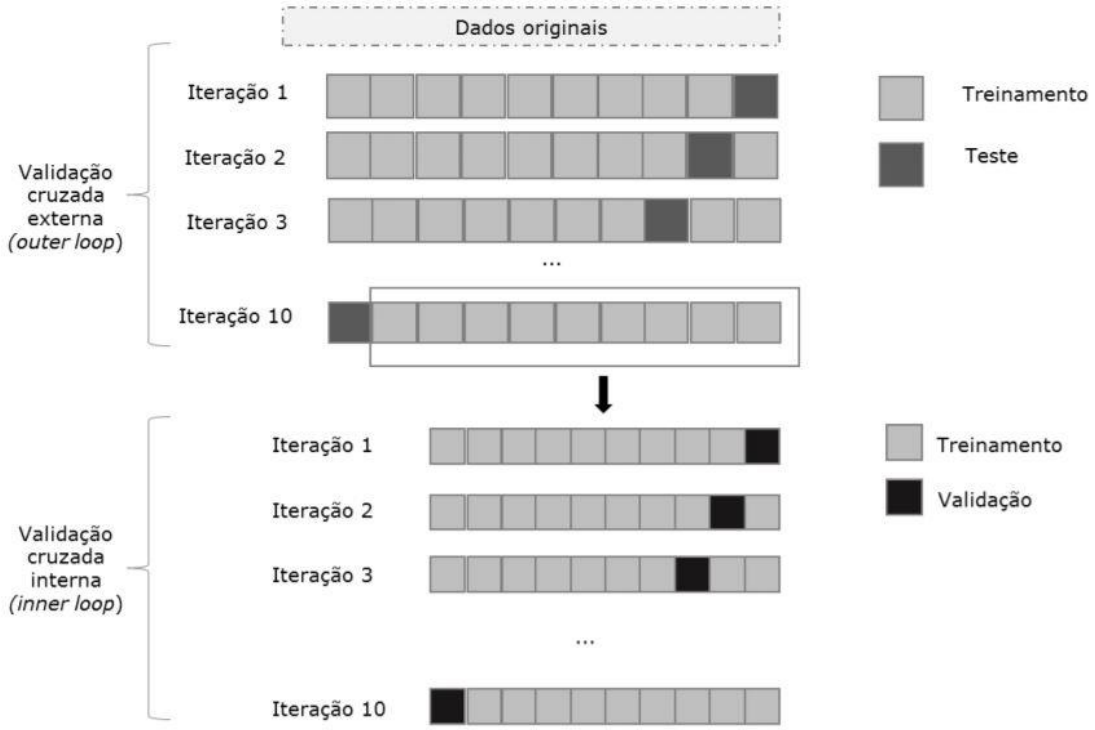

Figura 9 - Validação cruzada aninhada.
(RASCHKA, 2017)

A Figura 9 resume o processo de CV aninhado, onde k = 10, tanto para o CV interno quanto para o externo. No CV externo, o conjunto de dados foi dividido em 10 dobras e a cada iteração, 9 partes são usadas para treinamento e uma para teste. O loop do CV interno é realizado para cada iteração do CV externo. O conjunto de treinamento do CV interno também foi dividido em 10 partes e para as iterações de 1 a 10, 9 partes foram utilizadas no treinamento de modelos/ajuste de hiperparâmetros, e uma para estimar a performance preditiva.

2.4 Avaliação de modelos

Os resultados obtidos nesse trabalho serão avaliados por meio das métricas de precisão e cobertura (KENT, BERRY et al., 1955), além da matriz de confusão (PEARSON, 1904). Segundo SCIKIT-LEARN (2019), essas métricas são úteis quando se tem um conjunto de dados muito desequilibrado, além de serem as mais indicadas numa classificação binária.

2.4.1 Matriz de confusão

A matriz de confusão é uma ferramenta padrão para avaliação de modelos estatísticos. Trata-se de uma representação da qualidade obtida de uma classificação, expressada pela correlação de dados de referência com os dados classificados. A matriz de confusão é responsável por fornecer a base para criação de várias medidas de precisão e identificar os erros, auxiliando a refinar a classificação (FOODY, 2002).

A matriz de confusão é formada por um arranjo quadrado contendo as instâncias classificadas, dispostas em linhas e colunas que expressam o número de unidades de amostras de uma categoria.

		Classe predita	
		Não	Sim
Classe Atual	Não	Verdadeiro negativo	Falso positivo
	Sim	Falso negativo	Verdadeiro positivo

Figura 10 - Composição de uma matriz de confusão para classificação binária.
Traduzido de WITTEN et al. (2005)

A Figura 10 ilustra uma matriz de confusão de uma classificação binária, separada em quatro quadrantes:

- Verdadeiro positivo (*true positive, TP*) - representa a ocorrência de instâncias da classe positiva previstas pelo classificador corretamente. Por exemplo, quando uma pessoa está doente e o modelo previu corretamente que ela está doente.
- Falso positivo (*false positive, FP*) - representa a ocorrência de instâncias da classe positiva previstas pelo classificador incorretamente. Por exemplo, quando uma pessoa está saudável e o modelo previu erroneamente que ela está doente.

- Falso negativo (*false negative, FN*) - representa a ocorrência de instâncias da classe negativa previstas pelo classificador incorretamente. Por exemplo, quando uma pessoa está doente e o modelo previu erroneamente que ela está saudável.
- Verdadeiro negativo (*true negative, TN*) - representa a ocorrência de instâncias da classe negativa previstas pelo classificador corretamente. Por exemplo, quando uma pessoa está saudável e o modelo previu corretamente que ela está saudável.

Valores Reais	**Valores preditos**		
		Saudável (0)	Doente (1)
	Saudável (0)	TN = 583	FP = 25
	Doente (1)	FN = 5	TP = 28

Tabela 1 - Exemplo de matriz de confusão de uma classificação binária.

Interpretando a Matriz de confusão da Tabela 1, têm-se que de 33 pessoas doentes, 28 foram classificadas corretamente como doentes e 5 pessoas foram classificadas incorretamente como saudáveis. Já na classe negativa, composta por 608 pessoas saudáveis, 25 foram classificadas incorretamente como doentes e 583 foram classificadas corretamente como saudáveis. Observa-se ainda o desbalanceamento das classes previstas: a classe saudável (0), representa cerca de 95% das instâncias do conjunto de dados, enquanto que a classe doente representa apenas 5% das instâncias.

2.4.2 Precisão

A primeira das métricas derivadas da matriz de confusão usada neste trabalho é a precisão. Trata-se da razão entre os verdadeiros positivos e a soma de todas as previsões positivas, incluindo as falsas. A precisão é intuitivamente, a capacidade do

classificador de não rotular como positiva uma amostra que é negativa (SCIKIT-LEARN, 2019). Seu cálculo é representado pela fórmula:

$$Precisão = \frac{Verdadeiros\ positivos}{Verdadeiros\ positivos + Falsos\ positivos}$$

Para calcular a precisão do modelo ao classificar instâncias da classe positiva (doente), é preciso atentar para a disposição dos Verdadeiros positivos e Falsos positivos, de acordo com a Figura 11:

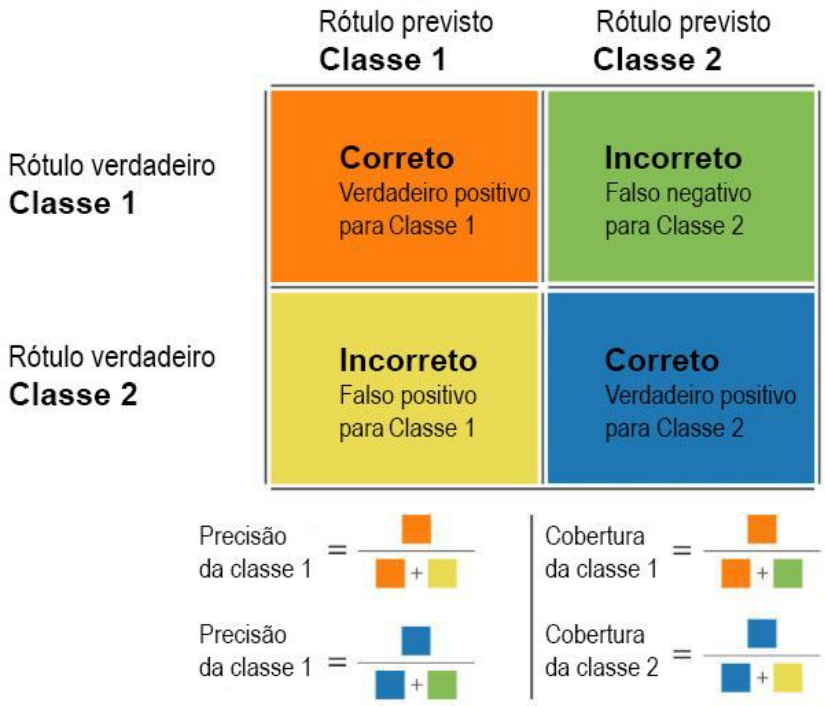

Figura 11 - Precisão e cobertura em uma classificação binária.
Traduzido de ROCCA (2019)

Logo, aplicando a fórmula ao exemplo da Tabela 2, têm-se que:

$$Precisão = \frac{28}{28 + 25} = 0{,}5283$$

Portanto, a precisão ao prever a classe positiva é de 52,83%.

2.4.3 Cobertura

A segunda métrica derivada da matriz de confusão é a cobertura. Trata-se da razão dos verdadeiros positivos e todas as previsões que realmente são positivas. A cobertura é intuitivamente, a capacidade do classificador de encontrar todas as amostras positivas (SCIKIT-LEARN, 2019). Esta métrica é representada pela seguinte fórmula:

$$Cobertura = \frac{Verdadeiros\ positivos}{Verdadeiros\ positivos + Falsos\ negativos}$$

Para a calcular a cobertura da classe positiva (doente), recorreu-se novamente a Figura 10, observando-se os valores dos Verdadeiros positivos e Falsos negativos. Com isso, aplicando a fórmula ao exemplo da Tabela 2, têm-se que:

$$Cobertura = \frac{28}{28 + 5} = 0{,}84$$

Assim, conclui-se que a cobertura ao classificar instâncias da classe positiva (doente) é de 84%.

2.4.4 Curva de precisão x cobertura

As curvas de precisão e cobertura são normalmente usadas na classificação binária para estudar a saída de um classificador. Uma curva de precisão e cobertura é um gráfico onde a precisão é plotada no eixo y e a cobertura é plotada no eixo x para diferentes limiares. Um classificador ideal seria capaz de alcançar a precisão perfeita de 1,0, que está relacionada a uma baixa taxa de falsos positivos. Esse mesmo classificador alcançaria a cobertura perfeita de 1,0, o que remete a uma baixa taxa de falsos negativos. Numa curva de precisão e recuperação, este ponto seria descrito pelo par ordenado (1, 1), localizado na parte superior à direita, de um gráfico de precisão e cobertura.

Com isso, é possível afirmar que, ao alcançar altas pontuações de precisão e cobertura, têm-se que o classificador está retornando resultados precisos (alta precisão), assim como retorna à maioria dos resultados positivos (alta cobertura), (SCIKIT-LEARN, 2019).

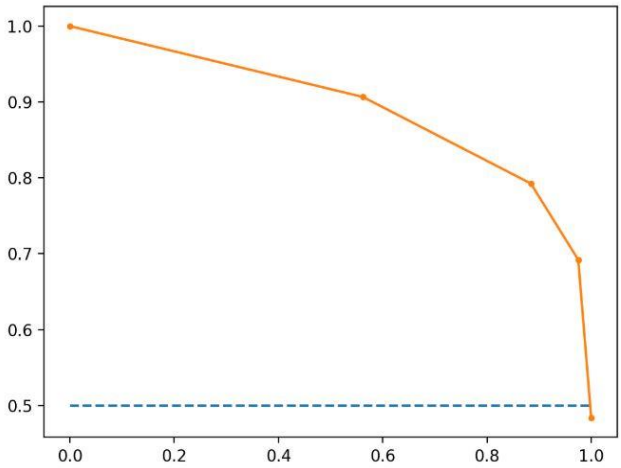

Figura 12 - Exemplo de uma curva de precisão e cobertura.
(BROWNLEE, 2018)

2.5 Trabalhos relacionados

Foram realizadas pesquisas em ambientes de busca de artigos acadêmicos onde encontrou-se dois experimentos relacionados com a identificação automática de potenciais clientes com maior propensão à compra dentre os leads obtidos por uma empresa através de estratégias de marketing digital por meio de algoritmos de classificação supervisionada. O primeiro artigo propõe uma previsão de vendas sob uma abordagem de regressão (YAN et al., 2012).

O artigo relata a problemática onde, tradicionalmente, os setores de vendas têm funcionado sem uma estratégia de vendas definida e sem uso de métricas para medir seu desempenho, o que ocasiona uma baixa eficiência, fazendo com que uma empresa demande muitos recursos para conseguir sucesso nas vendas.

A previsão de vendas sem estratégias é altamente complexa e a equipe de vendas tende a fazer um grande esforço, visto que com a possibilidade de perder alguma oportunidade, precisa-se entrar em contato todos os leads e classificá-los manualmente, sem falar do problema da possível falta de experiência de alguns vendedores, que podem superestimar ou subestimar a qualidade de alguns leads.

A abordagem de previsão de vendas do trabalho citado envolve duas etapas, aplicando o algoritmo de Regressão Logística para fazer a classificação. Primeiramente são coletados dados trimestrais da empresa, sendo criado um conjunto de treinamento, contendo atributos relevantes para a predição. A classificação possui três possíveis resultados: "vitória", para o caso de um lead estar propenso a tornar-se

um cliente e caso contrário, o lead pode ser rotulado com "derrota". Existe ainda o rótulo "pendente", quando o lead não foi classificado em nenhum dos dois outros rótulos. Para a segunda etapa, é feita a seleção do modelo que melhor classifica o rótulo "vitória" e, por fim, o modelo preditivo é avaliado através da métrica da pontuação de ganho. Com intuito de facilitar a interpretação da classificação pela equipe de vendas, foi desenvolvido um aplicativo que mostra os resultados e onde a equipe deve investir tempo.

O segundo artigo encontrado oferece uma modelagem probabilística de um funil de vendas para priorizar leads em empresas que usam softwares de automação de marketing (DUNCAN, ELKAN, 2015). A abordagem oferece uma alternativa a pontuação *"lead scoring*", gerada pelo software de aprendizado de máquina, através de cálculos estatísticos utilizados em algoritmos de aprendizado de máquina.

Usando o algoritmo de árvore de decisão e regressão, são apresentados modelos que fazem a classificação de leads com base em sua probabilidade de conversão para uma oportunidade de venda. Os modelos são treinados em cima de uma base gerada pelo software de automação de marketing que mantém registros de todos os leads, desde os frios, até os que foram convertidos em clientes, e que antes eram analisados um a um, manualmente, pelas equipes de marketing e de vendas. Para avaliar o desempenho dos métodos utilizou-se como métrica principal a acurácia, além de curvas de verdadeiros positivos e a curva de quais verdadeiros positivos foram convertidos em vendas reais.

Como resultado, a proposta classificou os leads em um tempo bem menor, fazendo com que a equipe de vendas se direcionasse para os leads mais quentes. Na empresa “C”, assim intitulada devido a um acordo de confidencialidade, com uma acurácia média de 97,6%, a taxa de conversão aumentou de 8% para 17% e o tempo médio para qualificar leads passou de 20 para 7 dias. Já a empresa “D”, que tinha uma taxa de conversão mais elevada, obteve-se um aumento de 31% para 37% e acurácia média de 96,25%. Os resultados mostraram ainda, como benefícios adicionais, a redução de esforço para qualificar leads e a redução de custos da empresa ao ter que realizar chamadas para agendar a demonstração do produto.

3 Experimentos

Neste Capítulo serão abordadas as etapas seguidas neste trabalho passo a passo, iniciando pela coleta da base de dados, pré-processamento, seleção de atributos, seleção de modelos, classificação e validação.

3.1 Base de dados

Para realização deste trabalho, foi necessário a obtenção de uma base de dados que continha dados reais de leads e clientes, assim como suas características, conforme proposto na Seção 2.2. A base foi concebida por meio de um acordo de confidencialidade entre a empresa e o autor deste trabalho e exportada através de um software de automação de marketing. Originalmente, possuía 6135 linhas, das quais 6075 correspondiam às instâncias rotuladas no atributo alvo como “lead” ou “lead qualificado” e 60 instâncias da classe “cliente”. Além disso, a base continha 34 colunas, contendo informações diversas como e-mail, nome, telefone, cargo, entre outras.

3.2 Pré-processamento

A fase de pré-processamento consiste no conhecimento e domínio da aplicação e preparação dos dados para a próxima fase. A base inicial estava no formato .xlsx e foi convertida para o formato .csv, compatível com as bibliotecas NumPy (NUMPY, 2019), SciPy (SCIPY, 2019), Pandas (PANDAS, 2019), Imblearn (IMBALANCED-LEARN, 2019), Matplotlib (MATPLOTLIB, 2019), Statistics (STATISTICS, 2019), IPython (IPYTHON, 2019) e Scikit-learn (SCIKIT-LEARN, 2019), esta última que contém os algoritmos necessários para realizar a classificação em métodos de aprendizado de máquina na linguagem de programação Python. Todas as bibliotecas acima estão disponíveis na distribuição gratuita de código aberto, Anaconda (ANACONDA, 2019).

Com intuito de promover uma melhoria no desempenho do classificador, foi necessário remover todas as colunas que continham dados faltosos, apagando as que

em qualquer elemento tivesse o valor “NaN”, reduzindo a quantidade de colunas para 11. Esse processo foi realizado fazendo uso da função “*dropna*” da biblioteca Pandas. A Figura 13 mostra como ficou a base de dados sem as colunas vazias. A coluna e-mail está em branco na imagem, devido ao acordo de confidencialidade.

	A	B	C	D	E	F	G	H	I	J	K
1	Email	Estágio no	URL públic	Perfil	Interess	Conversões	Data da prime	Origem da pri	Data da últir	Origem da últ	Eventos (Últimos 100
2		Lead	http://rdstati	d	0	4	8/15/16 13:10	Desconhecido	3/21/18 14:0	Desconhecido	Inativos - 60 dias / Rea
3		Lead	http://rdstati	d	0	2	8/15/16 13:10	Desconhecido	8/3/17 16:35	Desconhecido	Reativaram a conta / n
4		Lead	http://rdstati	d	0	2	8/15/16 13:10	Desconhecido	8/3/17 16:35	Desconhecido	Reativaram a conta / n
5		Lead Qualifi	http://rdstati	d	90	21	8/15/16 13:10	Desconhecido	4/17/18 14:5	Tráfego Direto	form_experimente / for
6		Lead	http://rdstati	d	0	2	8/15/16 13:10	Desconhecido	8/3/17 16:35	Desconhecido	Reativaram a conta / n
7		Lead	http://rdstati	d	0	4	8/15/16 13:10	Desconhecido	8/3/17 16:35	Desconhecido	Reativaram a conta / F
8		Lead	http://rdstati	d	0	2	8/15/16 13:10	Desconhecido	8/3/17 16:35	Desconhecido	Reativaram a conta / n
9		Lead	http://rdstati	d	0	2	8/15/16 13:10	Desconhecido	8/3/17 16:35	Desconhecido	Reativaram a conta / n
10		Lead	http://rdstati	d	0	2	8/15/16 13:10	Desconhecido	8/3/17 16:35	Desconhecido	Reativaram a conta / n
11		Lead	http://rdstati	d	0	2	8/15/16 13:10	Desconhecido	8/3/17 16:35	Desconhecido	Reativaram a conta / n
12		Lead	http://rdstati	d	0	2	8/15/16 13:10	Desconhecido	8/3/17 16:35	Desconhecido	Reativaram a conta / n
13		Lead	http://rdstati	d	0	4	8/15/16 13:10	Desconhecido	3/21/18 14:0	Desconhecido	Inativos - 60 dias / Rea
14		Lead	http://rdstati	d	0	2	8/15/16 13:10	Desconhecido	8/3/17 16:35	Desconhecido	Reativaram a conta / n
15		Lead	http://rdstati	d	50	6	8/15/16 13:10	Desconhecido	3/21/18 14:0	Desconhecido	Inativos - 60 dias / Cor

Figura 13 - Partícula extraída da base de dados com as colunas vazias apagadas.

3.2.1 Seleção de atributos

A próxima etapa foi a realização da seleção de atributos, conforme descrito na Seção 2.3.5. Esta técnica utilizou o método de visualização de dados com auxílio de um especialista que, ao fazer uma análise prévia dos dados, identificou os atributos relevantes usando o método de visualização dos dados. Observando os dados, o especialista identificou que os atributos relevantes seriam os elementos da coluna “Eventos (Últimos 100)”, que foi renomeada para “eventos”.

Estágio no funil	URL pública	Perfil - Lead Scoring	Interesse - Lead Scoring	Conversões	Data da primeira conversão	Origem da primeira conversão	Data da última conversão	Origem da última conversão	Eventos (Últim
Lead	http://rdstation.com.br/leads/public/1f416eb6-...	d	0	4	8/15/16 13:10	Desconhecido	3/21/18 14:02	Desconhecido	Inativos - 60 dias / Rea a conta
Lead	http://rdstation.com.br/leads/public/97588b52-...	d	0	2	8/15/16 13:10	Desconhecido	8/3/17 16:35	Desconhecido	Reativaram a members_export_51d7
Lead	http://rdstation.com.br/leads/public/f79cd83b-...	d	0	2	8/15/16 13:10	Desconhecido	8/3/17 16:35	Desconhecido	Reativaram a members_export_51d7
Lead Qualificado	http://rdstation.com.br/leads/public/2afcf8b9-...	d	90	21	8/15/16 13:10	Desconhecido	4/17/18 14:52	Tráfego Direto	form_experi form_experi forr
Lead	http://rdstation.com.br/leads/public/b2acde18-...	d	0	2	8/15/16 13:10	Desconhecido	8/3/17 16:35	Desconhecido	Reativaram a members_export_51d7
Lead	http://rdstation.com.br/leads/public/f1dfe9bc-...	d	0	4	8/15/16 13:10	Desconhecido	8/3/17 16:35	Desconhecido	Reativaram a Formulário de conta

Figura 14 - Partícula extraída da base de dados com 11 colunas.

Após a escolha dos atributos, foi feito um processo de binarização do atributo alvo que pode ser visto na Figura 14, representado pela coluna “Estágio no funil” (posteriormente renomeada para “estagio”), e responsável por rotular as instâncias como “Lead”, “Lead qualificado” ou “Cliente”. Para cada indivíduo classificado na base

de dados como “Lead” ou “Lead qualificado” atribuiu-se o valor “0” na célula correspondente (isto porque para o rótulo “Lead qualificado” ainda existia o risco do indivíduo não se tornar um cliente), e para cada “Cliente”, foi conferido o valor “1”. Esse processo foi realizado com a função “*map*” da biblioteca Pandas. Após isso, o atributo alvo foi realocado para a última coluna da base de dados.

	eventos	estagio
0	Inativos - 60 dias / Reativaram a conta / inat...	0
1	Reativaram a conta / members_export_51d7db9b76	0
2	Reativaram a conta / members_export_51d7db9b76	0
3	form_experimente / form_experimente / form_exp...	0
4	Reativaram a conta / members_export_51d7db9b76	0
5	Reativaram a conta / Formulário de contato do ...	0

Figura 15 - Partícula extraída da base de dados com o atributo alvo binarizado.

Prosseguindo, os elementos da coluna “eventos” estão indicados em formato de texto, estando os eventos de um lead separados por “/”, como pode-se observar na Figura 15. Estes elementos representam os eventos realizados por cada indivíduo desde seu primeiro contato com a empresa, como abertura de e-mails, cliques em links, download de conteúdo, entre outros. Após listar todos os eventos, foi sugerido pelo especialista, que fossem retirados os eventos que foram considerados irrelevantes para fazer a classificação, visto que se tratavam de eventos automáticos do sistema e não representavam interesse real por parte dos leads. Observou-se ainda que alguns dos demais eventos estavam sendo repetidos algumas vezes pelo mesmo indivíduo. Para cada um desses eventos, sem contar as repetições, foi criada uma nova coluna, ou seja, cada evento transformou-se numa característica da base de dados (coluna), que passou a conter 46 atributos + o atributo alvo. O preenchimento dos elementos desses atributos (cada célula na base de dados), foi feito obedecendo a condição de que, para cada indivíduo que tivesse realizado um determinado evento, fosse iniciado um contador. Ou seja, cada célula da base de dados foi preenchida inicialmente com o valor 0. A medida que os indivíduos realizavam um evento, o contador era iniciado, alterando o valor da célula para 1 e, no caso de eventos repetidos, o contador somava +1 para cada repetição. Ao final do pré-processamento, a base de dados ficou da seguinte maneira:

	e0	e1	e2	e3	e4	e5	e6	e7	e8	e9	...	e37	e38	e39	e40	e41	e42	e43	e44	e45	estagio
0	1	1	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	1	0	6	4	1	1	6	1	1	...	0	0	0	0	0	0	0	0	0	0
4	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
5	0	1	0	0	0	0	0	0	0	2	...	0	0	0	0	0	0	0	0	0	0

Figura 16 - Base de dados pré-processada.

Observe que o evento "e3" não ocorreu para a instância 1. Já o evento "e7" ocorreu 6 vezes para a instância 3.

3.3 Seleção de modelos

Como observado na Seção 3.2, a base de dados contém 6075 instâncias da classe “Lead” e apenas 60 da classe “Cliente”. Visando fazer o balanceamento da base de dados, o primeiro algoritmo a ser instanciado foi o SMOTE. Seu uso foi justificado na Seção 2.3.4, visto que foi necessário fazer uma sobreamostragem da classe minoritária “Cliente”, que representava menos de 1% das instâncias da base de dados. O segundo algoritmo instanciado foi o classificador. Como serão avaliados dois classificadores, Naive Bayes e Árvore de decisão, foram criados dois scripts, sendo um para cada classificador. Após isso, instanciou-se a classe Pipeline importada da biblioteca *Imblearn*, compatível com a técnica de sobreamostragem, SMOTE. Esta classe é responsável por encapsular os algoritmos usados para a seleção de modelos.

Para fazer a seleção de modelos, foi criado um dicionário contendo todos os hiperparâmetros a serem utilizados. Em todos os hiperparâmetros, foi utilizada a função “*randint*”, capaz de gerar números aleatórios entre um valor mínimo e um valor máximo determinados de acordo com o parâmetro. Inicialmente, adotou-se um conjunto de hiperparâmetros que posteriormente foi ajustado de acordo com uma observação minuciosa do comportamento de cada um deles e onde poderiam interferir na classificação.

O primeiro e único hiperparâmetro presente em ambos os scripts a ser analisado, está relacionado ao algoritmo SMOTE. Trata-se do número de vizinhos

mais próximos, representado por “*k_neighbors*”, utilizado para construir amostras sintéticas (LEMAITRE et al., 2016).

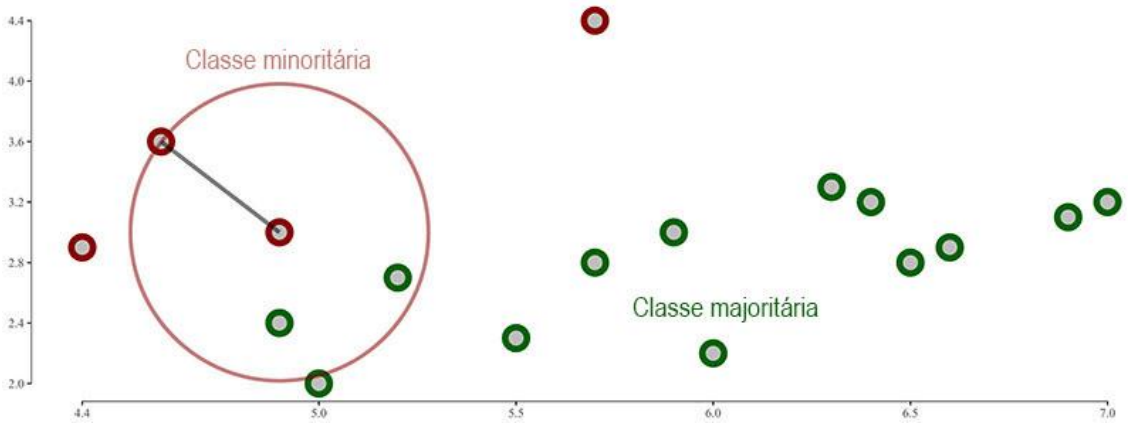

Figura 17 - Exemplo de sobreamostragem com SMOTE, onde k_neighbors = 1.
Traduzido de KUNERT (2017)

Na Figura 17, pode-se ver que, como *k_neighbors* = 1, apenas o vizinho mais próximo de k foi selecionado para gerar uma nova amostra. Para *k_neighbors* = 2, os dois vizinhos mais próximos são considerados, ou seja, para *k_neighbors* = n, os n vizinhos mais próximos são considerados para gerar amostras sintéticas.

Os demais hiperparâmetros são exclusivos do algoritmo Árvore de decisão, visto que o Naive Bayes não possui ajuste de parâmetros. O primeiro hiperparâmetro utilizado para fazer a seleção de modelos da Árvore de decisão foi o “*max_depth*”, que representa a profundidade máxima da árvore. O segundo hiperparâmetro ajustado foi o “*min_samples_split*”, o qual armazena o número mínimo de amostras necessárias para dividir um nó interno (ramo). Por fim, o último hiperparâmetro definido foi o “*min_samples_leaf*”, que especifica o número mínimo de amostras necessárias de um nó externo (folha; que quer dizer um nó sem filhos), (SCIKIT-LEARN, 2019). Por fim, tanto para a seleção de modelos, quanto para a classificação, utilizou-se como critério a entropia, explicado na Seção 2.3.2.

Inicialmente designou-se um conjunto de 4 hiperparâmetros observando sua metodologia de acordo com a biblioteca SCIKIT-LEARN. Observando a sensibilidade dos classificadores constatou-se que um conjunto extenso de valores de hiperparâmetros, prejudicava o desempenho do classificador ao predizer as amostras presentes na base de dados utilizada neste trabalho. Com isso, e a fim de evitar a problemática do sobre-ajuste, foi feito o ajuste do conjunto de valores dos hiperparâmetros, conforme apresentado na Tabela 2:

Hiperparâmetro	Valores (int)
N.º de vizinhos mais próximos	1 a 20
N.º mín. de amostras para dividir um nó externo	1 a 10
N.º mín. de amostras para dividir um nó interno	2 a 20
Profundidade máxima da árvore	2 a 20

Tabela 2 - Hiperparâmetros utilizados na seleção de modelos.

Com o conjunto de hiperparâmetros definido, utilizou-se a pesquisa aleatória (*Random search*) como técnica de busca para fazer a seleção de modelos. Conforme relatado na Seção 2.3.5, este método combina uma gama maior de hiperparâmetros, com mais chances de atingir o modelo ideal, utilizando o mesmo custo computacional que a abordagem de pesquisa em grade utilizaria para alcançar uma combinação em um conjunto de hiperparâmetros mais restrito. Esta escolha é justificada de acordo com seu artigo original. Segundo BERGSTRA e BENGIO (2012):

> "…para a maioria dos conjuntos de dados, apenas alguns dos hiperparâmetros realmente importam, mas diferentes hiperparâmetros são importantes em diferentes conjuntos de dados. Esse fenômeno faz com que a pesquisa em grade seja uma má escolha para a configuração de algoritmos para novos conjuntos de dados. "

Além disso, foram necessárias menos iterações do que se fosse utilizado o método de pesquisa em grade, ou seja, para encontrar o melhor modelo, foi necessário um "custo computacional" menor e que foi capaz de atingir uma gama maior de combinações de hiperparâmetros, como explicado na Seção 2.5.

Por fim, utilizou-se juntamente com a pesquisa randômica, a técnica da validação cruzada aninhada, vide Seção 2.4. O uso desta técnica é justificado a fim de que se possa estimar um desempenho de generalização imparcial. Quando a seleção de modelos é feita em um CV não aninhado, são usados os mesmos dados para ajustar os parâmetros do modelo e avaliar o desempenho. Com isso, as informações podem "vazar" para o modelo e ajustar os dados (SCIKIT-LEARN, 2019). Esse efeito depende, principalmente, do tamanho do conjunto de dados, que no caso deste trabalho, possui uma quantidade pequena de instâncias da classe positiva, *Cliente*.

3.4 Avaliação do modelo

De acordo com as técnicas apresentadas neste trabalho, pode-se observar que a tarefa de classificação se constitui da criação de um modelo classificador que utiliza um conjunto de dados conhecidos, objetivando estabelecer o valor de uma classe para instâncias não rotuladas (STEFANOWSKI, 2009). Com intuito de analisar a capacidade de predição dos modelos criados, foi escolhida primeiramente como métrica principal, a matriz de confusão, da qual derivam as demais métricas aplicadas neste trabalho, precisão e a cobertura, assim como a curva de precisão e cobertura. Todas as métricas foram previamente descritas na Seção 2.6.

A escolha se fundamenta visto que há um desequilíbrio no conjunto de dados, pois existem muitas ocorrências da classe majoritária que representa 99% das instâncias, enquanto existem poucas ocorrências da classe minoritária, que fica com o 1% restante. Segundo o SCIKIT-LEARN (2019), quando se têm uma grande quantidade de instâncias da classe majoritária, significa que se têm menos interesse em predizer corretamente esta classe. Analisando os cálculos de precisão e cobertura, observa-se que não é feito uso dos Verdadeiros negativos. Isto ocorre porque o modelo criado preocupa-se em predizer corretamente a classe minoritária, o qual esse trabalho se dedica. Quanto a escolha da curva de precisão e cobertura, SAITO e REHMSMEIER (2015) afirmam que:

> “O gráfico de cobertura e precisão é mais informativo do que o gráfico ROC ao avaliar classificadores binários em conjuntos de dados desbalanceados. ”

4 Resultados

Neste Capítulo serão apresentados os resultados obtidos de acordo as etapas propostas no Capítulo 3. Para cada classificador foram calculadas as métricas Matriz de confusão, que contém a quantidade de instâncias das classes positiva e negativa que foram classificadas corretamente e incorretamente; Precisão, que mostra o quão bem o modelo foi capaz de classificar corretamente uma classe, nesse caso a classe positiva (Cliente); Cobertura, que corresponde fração das instâncias classificadas em uma classe e que realmente fazem parte desta classe conforme seu rótulo verdadeiro e a Curva de precisão e cobertura, que remete a curva dos resultados de precisão e cobertura obtidos ao longo de 100 pesquisas aleatórias. Além disso, para o classificador Árvore de decisão, obteve-se ainda a árvore para os modelos que reportaram os maiores de índice de precisão e cobertura e o ranking de importância dos atributos. Ao final será feita uma análise dos resultados, além de uma comparação com os trabalhos relacionados.

4.1 Resultados com o classificador Árvore de decisão

A matriz de confusão do classificador Árvore de decisão apresentada na Tabela 3, reportou que 51 instâncias foram classificadas corretamente como “verdadeiras positivas” para a classe positiva (cliente, representada pelo valor 1) e 9 instâncias da mesma classe, classificadas incorretamente como “falsas negativas”. Para a classe negativa (lead, representada pelo valor 0), 5242 instâncias foram classificadas corretamente como “verdadeiras negativas” e 833 foram classificadas incorretamente como “falsas positivas”.

A Tabela 4 aponta uma precisão mínima, média e máxima com taxas de acerto de 4,6%, 5,8% e 8,4%, respectivamente e com desvio padrão de 0,7%. A cobertura apresentou índice mínimo, de 78,3%, médio de 85,8% e máximo de 91,7%, com desvio padrão de 2,5%.

Valores Reais	Valores preditos		
		Lead (0)	Cliente (1)
	Lead (0)	TN = 5242	FP = 833
	Cliente (1)	FN = 9	TP = 51

Tabela 3 - Matriz de confusão média.

	Mínima	Média	Máxima	Desvio padrão
Precisão	4,6%	5,8%	8,4%	0,7%
Cobertura	78,3%	85,8%	91,6%	2,5%

Tabela 4 - Valores de precisão e cobertura.

A métrica precisão mostrou o quão bem o modelo treinado classificou corretamente as instâncias da classe de interesse (cliente, 1), utilizando no seu cálculo apenas a coluna da classe prevista em questão. Como o conjunto de dados testado estava desbalanceado, visto que o SMOTE só foi aplicado no treinamento pois esteve encapsulado no pipeline (descrito na Seção 3.3), o número de amostras previstas incorretamente como sendo da classe minoritária (FP), é muito maior que o número de amostras previstas corretamente, isto porque tem-se 6075 amostras na classe majoritária e apenas 60 na minoritária. Por isso, a precisão média ao prever corretamente a classe minoritária no conjunto de dados desbalanceado foi de apenas 5,8%. Isso porque de 6130 instâncias, 884 foram classificadas na classe cliente, contudo, apenas 51 instâncias são realmente clientes. Apesar da baixa precisão, ela representa uma diminuição de esforço da equipe de vendas de cerca de 85%, se consideramos que sem o modelo todos os leads sejam contatados.

Já a cobertura, que leva em consideração na matriz de confusão apenas a linha da classe em questão, ou seja, seu Rótulo verdadeiro, representa o quão bem o modelo classificou corretamente as instâncias pertencentes a esta classe, portanto as que realmente fazem parte desta classe. Como são levadas em consideração apenas as instâncias reais da classe minoritária (cliente), a cobertura média encontrada foi de 85,8%. Ainda assim, foram "descartados" 9 clientes reais. Isso significa que, apesar

da concentração de esforços ter sido reduzida para apenas 15% das oportunidades, é fundamental ter uma equipe de vendas experiente, para que essas 9 oportunidades reais que seriam erroneamente descartadas pelo algoritmo árvore de decisão, não passem despercebidas pela equipe de vendas, evitando assim perda de receita da empresa.

Foi criado ainda, um gráfico (Figura 18), que mostra a variação dos valores de precisão e cobertura obtidos na seleção de modelos, retratando os desvios-padrões de 0,6% para precisão e 2,5% para cobertura, onde o eixo X representa as 100 pesquisas aleatórias e o eixo Y o índice obtido em cada métrica.

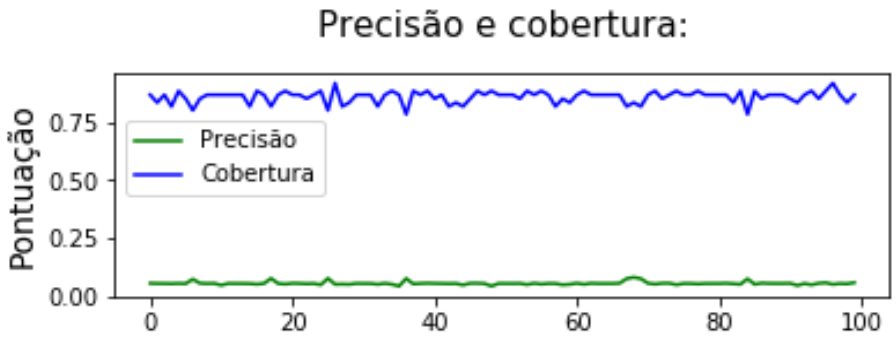

Figura 18 - Gráfico de variação de precisão e cobertura.

A curva de precisão e cobertura média obtida pelo classificador Árvore de decisão é apresentada na Figura 19, onde observou-se um pico inicial para o eixo de precisão, em função de uma cobertura com taxas entre 0 e 10%. A cobertura aumenta em função do decréscimo da taxa de precisão. Para uma precisão aproximada de 58%, têm-se uma cobertura de pouco mais de 20%. Quando a precisão atinge índices com acerto de cerca de 18%, a cobertura apresentada fica na casa dos 50%. O final da curva expressa o comportamento obtido de acordo com os resultados médios apresentados na Tabela 4, com uma precisão entre 4 e 9% e uma cobertura variando de 78,3% a 100%.

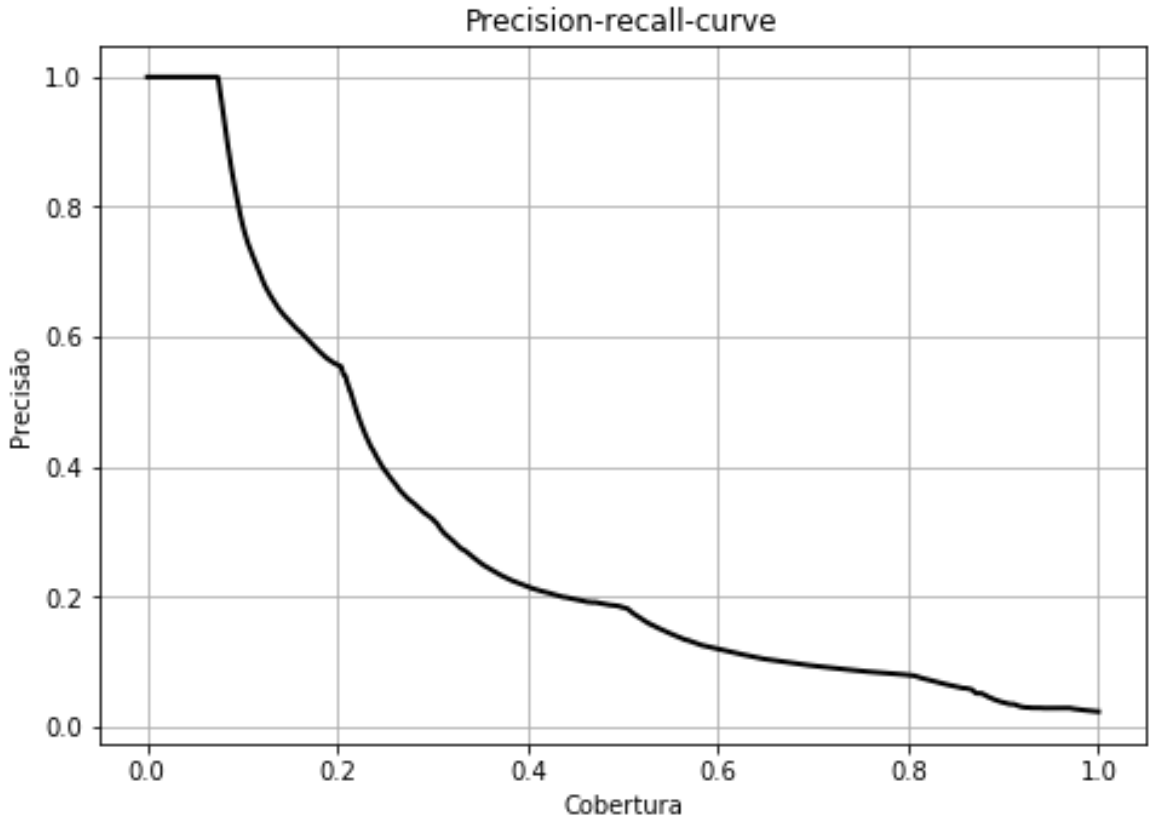

Figura 19 - Curva média de precisão e cobertura.

A curva de precisão e cobertura esteve bem distante do ponto ideal, que seria o valor 1 para o eixo x (cobertura), como também o valor 1 para o eixo y (precisão). Isto se deu devido ao fato de que as métricas foram obtidas em cima do conjunto de dados desbalanceado, que ocasionou uma baixa precisão, ou seja, a medida que a curva se aproximava do modelo ideal no eixo de cobertura, ela se distanciava do valor ideal para o eixo de precisão e vice-versa.

Foi extraído ainda o ranking de importância dos atributos para os modelos que apresentaram os melhores índices de precisão e cobertura. O ranking é elaborado de acordo com o ganho de informação de cada atributo, conforme descrito na Seção 2.3.2, indicando a importância dos atributos. Essa informação é de grande importância para a empresa, visto que os atributos representam os eventos realizados, ou seja, é de interesse da empresa saber quais os principais eventos que ajudam na classificação. Os atributos tiveram seus nomes originais substituídos, devido o acordo de confidencialidade. Eles estão numerados conforme a ordem em que foram encontrados na coluna eventos.

Na Tabela 5 é apresentado o ranking de importância dos atributos obtido a partir do modelo que reportou a maior taxa de precisão. É possível observar que o atributo e12 teve o maior ganho de informação, enquanto o atributo e1, obteve o menor ganho listado. No total, 14 atributos apresentaram ganho de informação significativo. Os atributos ausentes tiveram ganho de informação próximo de 0.

Atributo	Ganho
e12	0.291505%
e14	0.259833%
e18	0.137491%
e0	0.107021%
e3	0.092801%
e25	0.023829%
e45	0.016897%
e7	0.015263%
e15	0.013970%
e36	0.010499%
e4	0.009860%
e16	0.006741%
e11	0.005795%
e30	0.003656%

Tabela 5 - Ranking de importância para o modelo de maior precisão.

A Figura 20 representa a Árvore de decisão gerada a partir do modelo com maior precisão onde N.º de vizinhos mais próximos = 5; N.º mín. de amostras para dividir um nó externo = 1; N.º mín. de amostras para dividir um nó interno = 18 e Profundidade máxima da árvore = 9. Pode-se ver que a árvore leva em consideração o critério entropia, apresentado na Seção 2.3.2. As divisões da árvore são geradas de acordo com o hiperparâmetros presentes no modelo. A árvore apresentada informa apenas até a sua 5ª divisão, visto que a árvore completa gerou uma imagem que ficou ilegível ao para visualização neste trabalho. É possível visualizar a árvore completa em: https://i.ibb.co/gj6QRyZ/rvore-maior-precis-o.png.

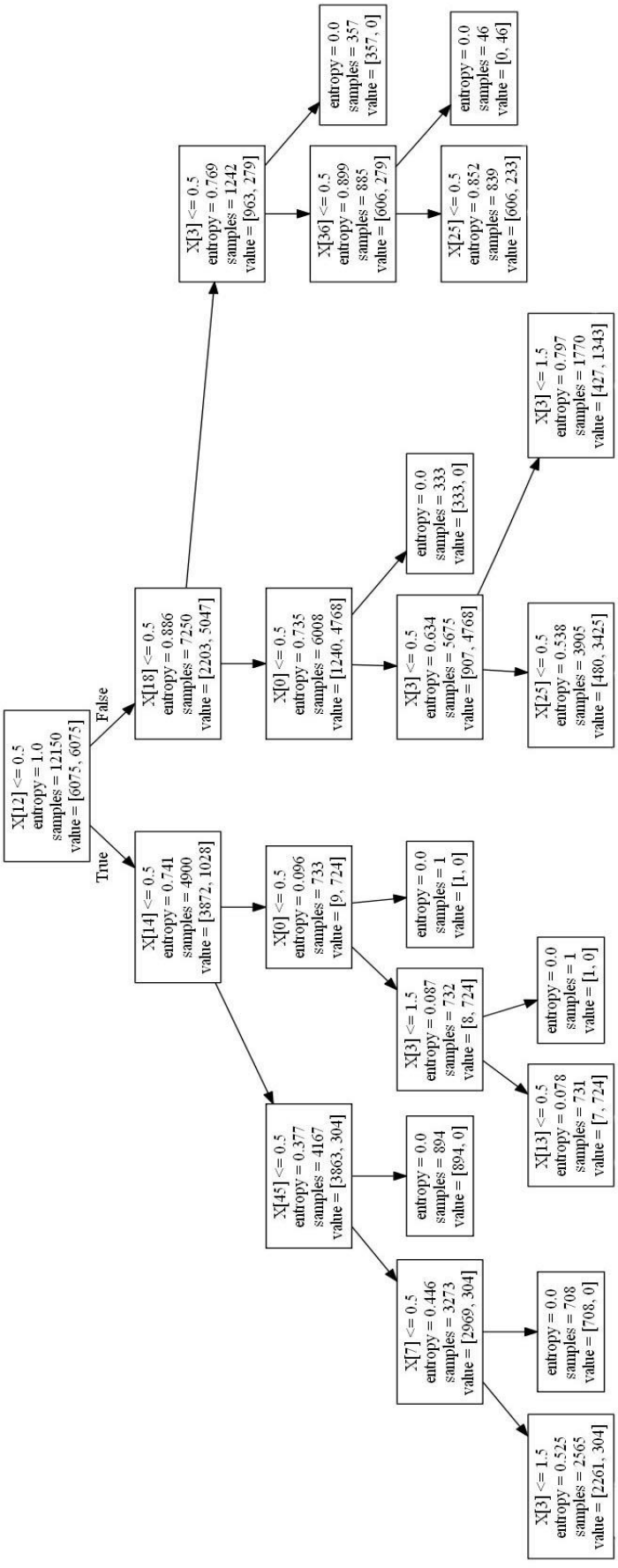

Figura 20 - Árvore de decisão para o modelo com maior precisão.

A seguir, extraiu-se o ranking de importância dos atributos a partir do modelo onde obteve-se o melhor resultado para a métrica cobertura. O ranking está apresentado na Tabela 6. Observa-se que o atributo e14 obteve o maior ganho de informação, enquanto o atributo e36 reportou o menor. É possível ver que apenas 9 atributos tiveram ganho de informação. Os demais atributos tiveram ganho de informação nulo, portanto não houve necessidade de incluí-los neste ranking.

Atributo	Ganho
e14	0.297388%
e12	0.296141%
e18	0.147086%
e0	0.116939%
e3	0.082281%
e25	0.017014%
e7	0.015902%
e45	0.015558%
e36	0.011691%

Tabela 6 - Ranking de importância de atributos.

Por fim, na Figura 21 é apresentada a Árvore de decisão para o modelo que reportou o maior índice da métrica cobertura. Assim como na Figura 22, as divisões desta árvore levam em consideração o critério entropia e os hiperparâmetros do modelo em questão.

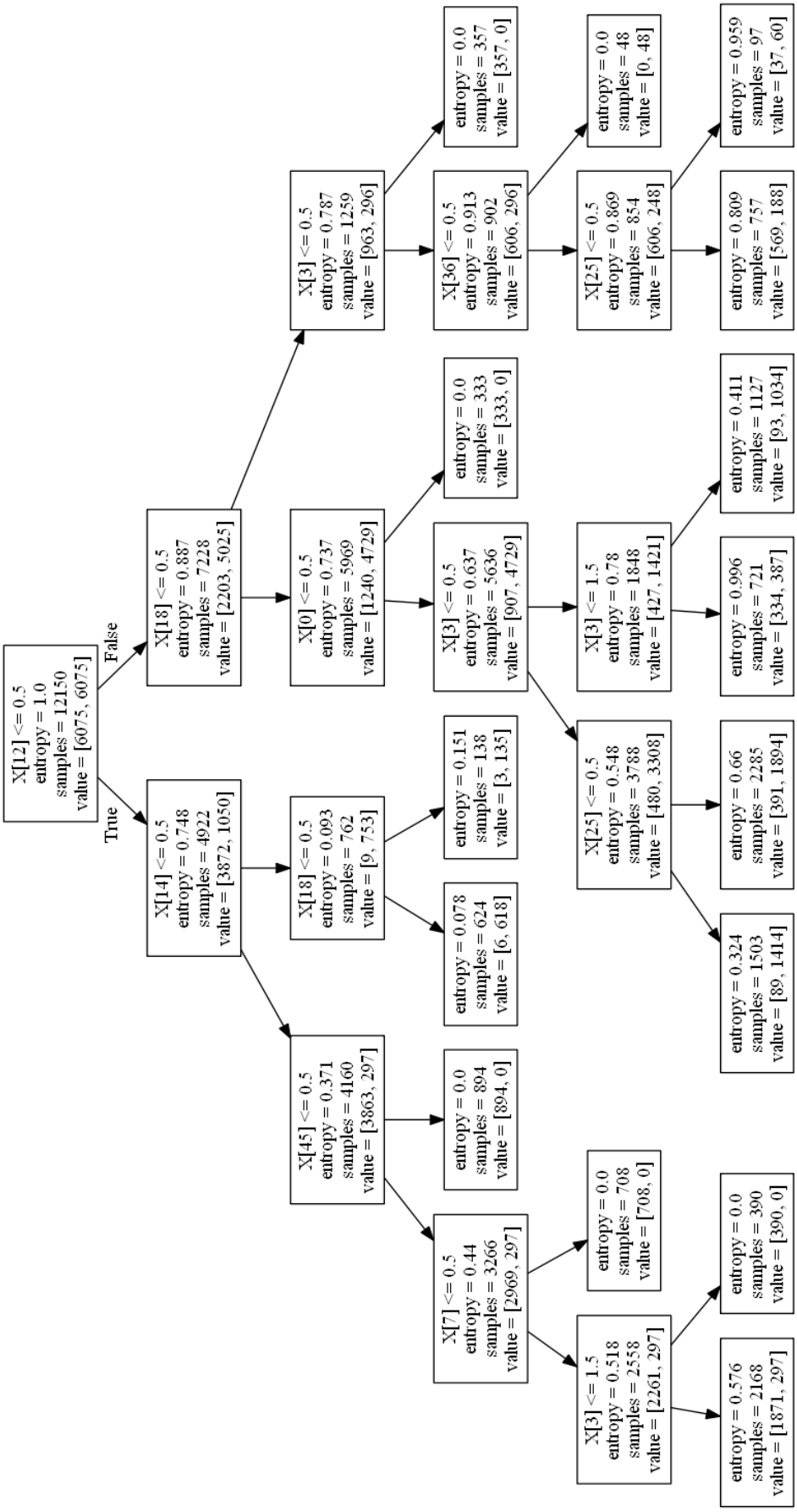

Figura 21 - Árvore de decisão para o modelo com maior cobertura.

4.2 Resultados com o classificador Naive Bayes

A matriz de confusão média reportada pelo classificador Naive Bayes é apresentada na Tabela 7. É possível ver que os resultados estão bem próximos ao classificador Árvore de decisão: de 60 instâncias da classe de interesse (cliente, 1), 49 foram classificadas corretamente (verdadeiras positivas, TP) e 11 instâncias foram classificadas incorretamente como "falsas negativas" (FN). Na classe negativa (lead, 0), o resultado foi de 5150 instâncias classificadas corretamente (verdadeiras negativas) e 925 foram classificadas incorretamente como "falsas positivas".

A Tabela 8 reporta os índices de precisão mínima, média e máxima, com taxas respectivas de 4,9%, 4,9% e 5,1%. Essa baixa variação apontou um desvio padrão de apenas 0,02%. Na métrica cobertura, obteve-se nos índices mínimo e médio uma taxa de 80%. O índice máximo encontrado foi de 83,3% e o desvio padrão foi de 0,4%.

Valores Reais	**Valores preditos**		
		Lead (0)	Cliente (1)
	Lead (0)	TN = 5150	FP = 925
	Cliente (1)	FN = 11	TP = 49

Tabela 7 - Matriz de confusão média para o classificador Naive Bayes.

	Mínima	**Média**	**Máxima**	**Desvio padrão**
Precisão	4,9%	4,9%	5,1%	0,02%
Cobertura	80%	80%	83,3%	0,4%

Tabela 8 - Valores de precisão e cobertura.

A precisão média obtida pelo classificador Naive Bayes foi de 4,9%, que se mostrou próxima do valor obtido no classificador Árvore de decisão (menos de 1% de diferença). Novamente obteve-se um valor baixo devido ao cálculo desta métrica levar em consideração a coluna com os valores preditos para a classe positiva, o que

resultou numa proporção entre os valores corretos da classe minoritária e a soma deles com os valores incorretos da classe majoritária.

A cobertura média encontrada foi de 80%, contra 85,8 do classificador Árvore de decisão. Isso porque, diferentemente da precisão, a cobertura não leva em consideração a classe majoritária. Seu cálculo refletiu a proporção das instâncias da classe positiva (cliente) que foram realmente classificadas como clientes.

Assim como no classificador Árvore de decisão, foi gerado um gráfico que expressa a variação destas métricas, obtidas durante a execução das pesquisas aleatórias (Figura 22), onde o eixo X representa as pesquisas aleatórias e o eixo Y os valores de precisão e cobertura. O gráfico é fiel aos valores de desvio padrão encontrados: 0,02% para a precisão e 0,4% para a cobertura.

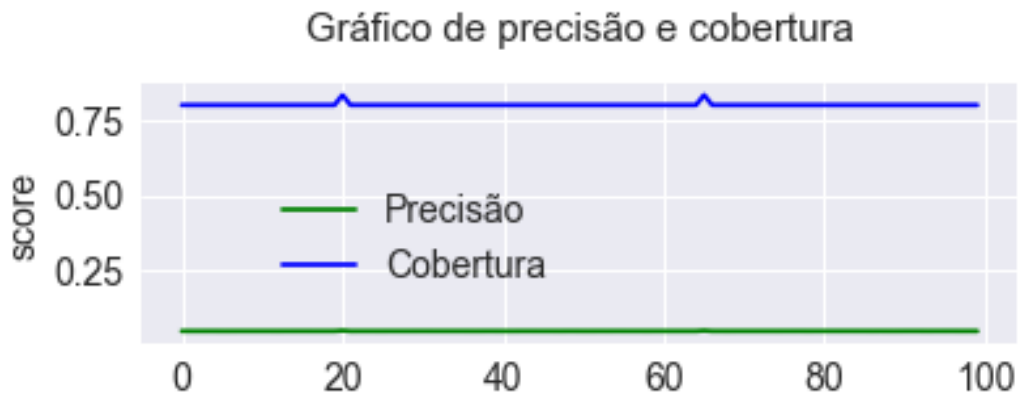

Figura 22 - Gráfico dos valores de precisão e cobertura durante pesquisas aleatórias.

A Figura 23 representa o gráfico da curva de precisão e cobertura média, obtida a partir das métricas de precisão e cobertura do classificador Naive Bayes. Observa-se o comum pico inicial para o eixo de precisão com uma taxa de 100%, para uma cobertura próxima de 0%. Logo após a outro pico, com precisão em torno de 40% e cobertura entre 0% e 5%. Por fim, o desempenho do classificador torna-se mais regular, com uma precisão variando entre 5% e 10% e uma cobertura indo de aproximadamente 5% até a taxa de 80%. Essa parte da curva representou bem o desempenho médio encontrado nas métricas de precisão e cobertura. Por fim, encontrou-se uma precisão decrescente e entre 5% e 1% para uma cobertura variando de 81% a 100%.

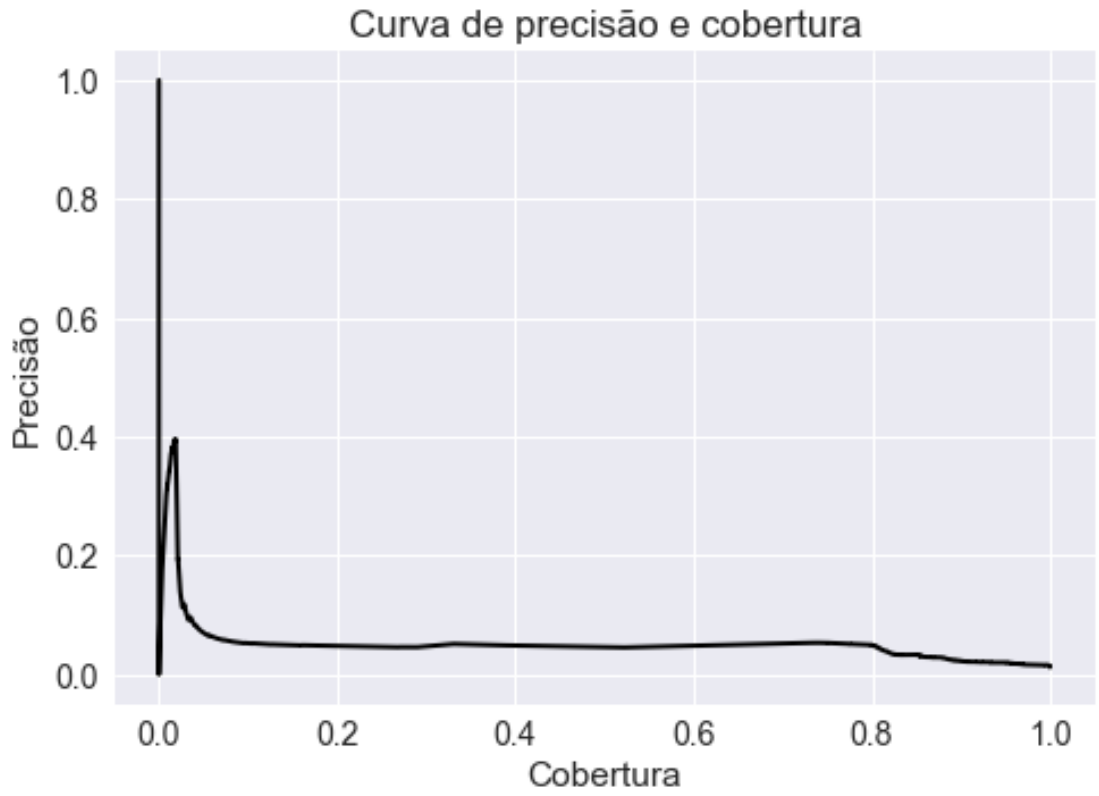

Figura 23 - Curva de precisão e cobertura média.

Analisando a curva de precisão e cobertura média do classificador Naive Bayes, observa-se que, à medida que a curva se aproximava do ponto ideal para a métrica do eixo de cobertura, ela se distanciava do ponto ideal para o eixo correspondente a precisão, refletindo os resultados médios encontrados no cálculo dessas métricas.

Se este classificador fosse o único com suas métricas calculadas, a redução de trabalho da equipe de vendas seria de cerca de 81,2% pois 5161 instâncias foram classificadas como lead. Os esforços seriam destinados aos outros 18,8%, classificados como clientes. Se a equipe de vendas levasse em consideração apenas os resultados obtidos pelo classificador Naive Bayes, haveria uma redução drástica nos recursos dedicados às reais oportunidades. Entretanto, haveria uma perda de receita pela classificação incorreta de 11 instâncias da classe positiva. Por isso, novamente, enfatiza-se que se faz necessário uma equipe de vendas experiente que seja capaz de observar as oportunidades além do que foi reportado pelo classificador.

4.3 Comparação com trabalhos relacionados

Ambos os trabalhos relacionados apresentaram similaridade ao objetivo maior deste trabalho, entretanto, foram realizados experimentos, algoritmos e métricas diferentes das utilizadas neste trabalho. Apesar disso, pode-se afirmar que, a partir dos resultados mostrados nos 3 trabalhos, a aplicação de algoritmos de aprendizado supervisionado em conjuntos de dados obtidos a partir de softwares de automação de marketing digital, teria grande importância se fosse praticada comumente entre empresas que utilizam este tipo de marketing. Isso porque mostrou-se uma possível diminuição da carga sobre as equipes de vendas, possibilitando inúmeros benefícios para as empresas, como a redução de custos ao investir tempo apenas nas reais oportunidades, o que geraria um aumento de receita mais rápido, liberando a equipe de vendas para trabalhar em novas oportunidades.

5 Conclusão

Este trabalho teve como objetivo principal a identificação de potenciais clientes com maior propensão à compra dentre os leads obtidos por uma empresa através de estratégias de marketing digital. Isso possibilita uma redução considerável na carga imposta para as equipes de vendas, que passará a trabalhar diretamente em cima das oportunidades com maior probabilidade de compra. Para isso, analisou-se os resultados obtidos a partir dos classificadores de aprendizado supervisionado, Árvore de decisão e Naive Bayes.

Os resultados descritos no Capítulo anterior reportaram uma taxa média máxima encontrada para a métrica precisão de 5,8%. Este valor representou a proporção de amostras positivas classificadas corretamente, ou seja, das 884 instâncias classificadas como positivas, apenas 51 são realmente amostras da classe positiva. Na prática, isso quer dizer que se apenas essa métrica fosse avaliada, apenas uma pequena quantidade das instâncias classificadas como positivas seriam de fato, convertidas em clientes. Isso mostrou a importância de se ter um conjunto de dados balanceado quando se visa obter métricas que relacionam as amostras preditas em classes distintas. Apesar dos valores apresentados para esta métrica, vale salientar que o total de amostras que demandam esforços para a equipe de vendas foi reduzido de 6130 para 884.

Na métrica cobertura, que representa a proporção dos positivos identificados corretamente, foi encontrada uma taxa média máxima de 85,8%. Levando para o dia-a-dia da empresa, entende-se que através da obtenção desta métrica, a classificação supervisionada identificou que das 60 amostras da classe cliente, 51 foram corretamente classificadas e seriam passadas para a equipe de vendas. Com isso, é possível afirmar que o classificador identificou corretamente a maior parte dos potenciais clientes, o que é de grande importância para a empresa.

A conclusão final dos resultados obtidos é de que a métrica de maior importância para alcançar o principal objetivo deste trabalho foi a cobertura, visto que esta métrica informou o quão bom o modelo se saiu para detectar amostras da classe positiva, ou seja, para identificar os potenciais clientes. Com isso pode-se dizer, que devido às altas taxas encontradas os resultados são animadores e comprovam a importância da utilização do aprendizado de máquina quando aplicado junto ao

objetivo maior deste trabalho: através da identificação dos rótulos das classes lead e cliente, foi constatado pelo classificador Árvore de decisão, (que obteve os melhores resultados), que 85% das amostras do conjunto de dados não tinha interesse real no produto/serviço ofertado pela empresa, o que significa uma redução de grande impacto sobre os esforços emitidos pela equipe de vendas. Observou-se ainda que, se fosse levado em consideração apenas os resultados encontrados, poderia ocorrer uma possível perda de clientes, visto que o modelo classificou incorretamente 9 amostras da classe cliente como sendo pertencente a classe lead. A classificação reconheceu que a equipe de vendas deveria levar em consideração as amostras pertencentes aos 15% restantes do conjunto de dados. Com isso, ressalta-se novamente a importância de uma equipe de vendas experiente, que seja capaz de identificar oportunidades classificadas incorretamente pelo modelo, evitando uma possível perda de receita para a empresa. Por fim, o classificador Árvore de decisão apresentou ainda um ranking de importância de atributos, informação extremamente útil para a empresa, pois através deste ranking foi possível extrair quais os eventos mais relevantes para identificar corretamente as amostras da classe cliente.

Apesar de atingir parcialmente seu objetivo maior, foram encontradas algumas limitações ao elaborar este trabalho. A maior delas pode estar relacionada com o fato de não ter sido feita uma escolha mais criteriosa dos algoritmos de classificação supervisionada utilizados. Outra dificuldade pode ser relacionada ao desbalanceamento do conjunto de dados durante a obtenção da métrica precisão, resultando em índices desfavoráveis. Outra limitação é reportada pela utilização de apenas um conjunto de dados durante todos os experimentos deste trabalho. Para que seja possível utilizar outros conjuntos de dados, existe a limitação de que eles deverão conter colunas iguais às do conjunto utilizado neste trabalho. Com isso, pode-se afirmar que a observação e melhoria destas questões podem enriquecer a continuação deste trabalho.

A análise dos resultados expressa ainda a notória importância da continuidade do desenvolvimento do objetivo maior deste trabalho. Então, para trabalhos futuros sugere-se a obtenção de novos conjuntos de dados para ser utilizados como conjuntos de testes para predição. Também, a utilização de outros classificadores enriqueceria o trabalho, o que daria mais solidez aos resultados. Por fim, poderia ser descrita uma comparação envolvendo os custos da empresa ao ter que realizar os contatos antes da classificação e após a classificação, mostrando na prática o impacto

da aplicação do aprendizado supervisionado para empresas que utilizam softwares de automação de marketing digital.

Referências

ANACONDA. Anaconda Distribution. (2019). The World's Most Popular Python/R Data Science Platform. Disponível em https://www.anaconda.com/distribution/.

BERGSTRA, James; BENGIO, Yoshua. (2012). Random Search for Hyper-Parameter Optimization. Département d'Informatique et de recherche opérationnelle. Université de Montréal. Montreal, QC, H3C 3J7, Canadá.

BISHOP, C. M. (2006). Pattern Recognition and Machine Learning. [S.l.]: Springer.

BREIMAN, L., FRIEDMAN, J. H., OLSHEN, R. A., & STONE, C. J. (1984). Classification and Regression Trees. Wadsworth.

BROWNLEE, Jason. (2018). How and When to Use ROC Curves and Precision-Recall Curves for Classification in Python. Disponível em: https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/.

CANDIAGO, Lorenzo. (2017). Algoritmo de Classificação Naive Bayes. Disponível em: https://www.organicadigital.com/seeds/algoritmo-de-classificacao-naive-bayes/

CARO, Abrão. (2010). Comportamento do Consumidor e a Compra Online: uma análise multicultural. Tese (Pós-Graduação) - Universidade de São Paulo.

CAWLEY, G.C.; TALBOT, N.L.C. (2010,11). On over-fitting in model selection and subsequent selection bias in performance evaluation. J. Mach. Learn. Res. 2079-2107.

CHAWLA, N. V., BOWYER, K. W., HALL, L. O., & KEGELMEYER, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. J. Artif. Intell. Res. (JAIR) 16:341–378.

DANG, Xuan T., HIROSE, Osamu, SAETHANG, Thammakorn, TRAN, Vu Anh, NGUYEN, Lan Anh, LE, Tu Kien T., KUBO, Mamoru , YAMADA, Yoichi, SATOU, Kenji. (2013). A novel over-sampling method and its application to miRNA prediction. J. Biomedical Science and Engineering, 6, 236-248.

DUDA, Richard O.; HART, Peter E; STORK, David G. (2001). Pattern classification. 2nd ed. New York, N.Y: John Wiley & Sons.

DUNCAN, Brendan, ELKAN, Charles. (2015). Probabilistic Modeling of a Sales Funnel to Prioritize Leads. Department of Computer Science and Engineering, University of California, San Diego, USA.

FONSECA, J. (1994). Indução de árvores de decisão. Tese de Mestrado, Lisboa.

FOODY, G. M. (2002). Status of land cover classification accuracy assessment. Remote Sensing of Environment, v. 80, p. 185– 201.

GAMA, J. (2000). Árvores de Decisão. Disponível em: http://www.liacc.up.pt/~jgama/Mestrado/ECD1/Arvores.html.

GARCIA, G.M. (2007). Comportamento do consumidor virtual: a influência das características pessoais na intenção de compra. Tese (Pós-Graduação) - Universidade Federal do Rio Grande do Sul.

GOEBEL M., GRUENWALD, L. (1999). A Survey of Data Mining and Knowledge Discovery Software Tools. ACM SIGKDD Explorations, New York, v. 1, no. 1, p. 20-33.

GOMES, L.; GOMES, C.; ALMEIDA, A. (2002). Tomada de decisão gerencial. São Paulo: Atlas.

GONZALEZ, R. C.; WOODS, R. E. (2002). Digital Image Processing. 2. ed. Upper Saddle River (NJ, USA): Prentice-Hall.

HE, H.; GARCIA, E. A. (2009). Learning from Imbalanced Data. IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, [S.l.], v.21, n.9, p.1263–1284.

IMBALANCED-LEARN. (2019). Disponível em https://imbalanced-learn.readthedocs.io/en/stable/index.html.

IPYTHON. (2019). IP[y]: IPython Interactive Computing. Disponível em http://ipython.org/.

KENT, A.; BERRY, M.; LUEHRS, F. U.; & PERRY, J. W. (1955). Machine literature searching. VIII: operational criteria for designing information retrieval systems, American Documentation, issue VI, vol 2, pp. 93-101.

KOTLER, Philip. (2010). Marketing 3.0. Tradução de Ana Beatriz Rodrigues. Rio de Janeiro: Elsevier.

KOTLER, Philip. (1996). Marketing: edição compacta. Ed. compacta. São Paulo: Atlas.

KUNERT, Richard. (2017). SMOTE explained for noobs - Synthetic Minority Over-sampling TEchnique line by line. Disponível em: http://rikunert.com/SMOTE_explained.

LATYSHEVA, Natasha; RAVARANI, Charles. (2016). Expanding your machine learning toolkit: Randomized search, computational budgets, and new algorithms. Disponível em: https://cambridgecoding.wordpress.com/2016/05/16/expanding-your-machine-learning-toolkit-randomized-search-computational-budgets-and-new-algorithms-2/.

LEMAITRE, G.; NOGUEIRA, F.; OLIVEIRA, D.; ARIDAS, C. (2016). Imblearn.over_sampling.SMOTE.https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over_sampling.SMOTE.html.

LENINE, F. (2017). Como selecionar atributos para resolver a maldição da dimensionalidade. Disponível em https://medium.com/@fabiolenine/como-selecionar-atributos-para-resolver-a-maldi%C3%A7%C3%A3o-da-dimensionalidade-5c810bc8449f.

LIU, H.; YU, L. (2002). Feature selection for data mining. Disponível em: http://www.public.asu.edu/~huanliu/feature_selection.html.

MATPLOTLIB. (2019). Disponível em https://matplotlib.org/.

MICHALSKI, R. S.; BRATKO, I.; KUBAT, M.; editors. (1998). Machine Learning and Data Mining: Methods and Applications. John Wiley and Sons. West Sussex, England.

MITCHELL, T.M. (1997). Machine Learning, Pittsburgh: McGraw Hill.

NUMPY. (2019). Disponível em https://www.numpy.org/.

PANDAS. (2019). Python Data Analysis Library. Disponível em http://pandas.pydata.org/.

PEARSON, Karl F. R. S. (1904). Mathematical contributions to the theory of evolution. XIII. On the theory of contingency and its relation to association and normal correlation. Biometric series, I. University College, University of London.

PIRAMUTHU, S. (2006). On preprocessing data for financial credit risk evaluation. Expert Systems with Application, 30(3), 489-497.

QUINLAN, J. R. (1993). C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

QUINLAN, J. R. (1979). Discovering rules by induction from large collections of examples. In D. Michie (Ed.), Expert systems in the microelectronic Age. Edinburgh University Press.

RASCHKA, S. (2015). Python Machine Learning. 2. ed. Birmingham: Packt Publishing Ltd.

REBOUÇAS, P.S.M.D. (2011). Metodologias de Classificação Supervisionada para Análise de Dados de Microarrays." Doutorado em Estatística e Investigação Operacional (Especialidade de Probabilidades e Estatística).

RESULTADOS DIGITAIS. (2016). *Inbound* Marketing. Disponível em: https://resultadosdigitais.com.br/inbound-marketing/.

RESULTADOS DIGITAIS. (2015). Lead Scoring: o guia definitivo. Disponível em: https://resultadosdigitais.com.br/blog/lead-scoring/.

ROCCA, Baptiste. (2019). Handling imbalanced datasets in machine learning. Disponível em: https://towardsdatascience.com/handling-imbalanced-datasets-in-machine-learning-7a0e84220f28

RODRIGUES, F. A. (2016). Métodos de redução de dimensionalidade e seleção de atributos. Depto. de Matemática Aplicada e Estatística - SME, Instituo de Ciências Matemáticas e de Computação, USP.

RUSS, J. C. (1990). Computer-Assisted Microscopy: The Measurement and Analysis of Images. New York (NY, USA): Plenum Publishing Corporatio.

SAITO, Takaya; REHMSMEIER, Marc. (2015). The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. Disponível em https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0118432.

SCIKIT-LEARN. (2019). Machine Learning in Python. Disponível em: https://scikit-learn.org/stable/index.html.

SCIPY. (2019). SciPy library. Diponível em https://www.scipy.org/scipylib/index.html.

SEGURA, M. C. (2009). O estudo do Marketing Digital versus Marketing Tradicional e a percepção das suas campanhas por parte dos consumidores no mercado virtual a tradicional. Tese (Mestrado em estatística e Gestão da Informação) - Instituto Superior de Estatística e Gestão de Informação da Universidade Nova de Lisboa, Lisboa.

SOLOMON, MICHAEL R. (2011). O comportamento do consumidor: comprando, possuindo e sendo. 9. ed. Porto Alegre, RS: Bookman.

SOUZA, BRUNO. (2012). Marketing Digital 2.0: como sair na frente da concorrência. [ebook] Disponível em: http://www.gestordemarketing.com

STATISTICS. (2019). Statistics - Mathematical statistics functions. Disponível em https://docs.python.org/3/library/statistics.html.

STEFANOWSKI, J. (2009). Data Mining: Evaluation of Classifiers. Disponível em: http://www.cs.put.poznan.pl/jstefanowski/sed/DM-4-evaluatingclassifiersnew.pdf.

STEINER, Maria Teresinha Arns; SOMA, Nei Yoshihiro; SHIMIZU, Tamio; NIEVOLA, Júlio Cesar; LOPES, Fábio Mendonça; SMIDERLE, Andréia. (2004). Redes neurais e árvores de decisão na análise do crédito bancário. São João del-Rei, MG. Publicado no XXXVI - SBPO.

THEODORIDIS, S.; KOUTROUMBAS, K. (2003). Pattern Recognition. 2nd Edition, Academic Press, San Diego.

TOLEDO, A.L; CAMPOMAR, C. M; TOLEDO, L. D. (2006). Planejamento de marketing e Confecção do Plano de Marketing: Uma Análise Crítica, São Paulo, v.13, n.37, abr./jun.

TORRES, CLÁUDIO. (2009). A Bíblia do Marketing Digital. São Paulo: Novatec Editora.

VAZ, C. A. (2010). Google Marketing: o guia definitivo de marketing digital. São Paulo: Novatec.

WITTEN, I. H. (IAN H.); FRANK, EIBE. (2005). Data mining : practical machine learning tools and techniques. 2rd ed. p. cm. – (Morgan Kaufmann series in data management systems).

WITTEN, I. H.; FRANK, E.; HALL, M. A.; PAL, C. J. (2016). Data mining : practical machine learning tools and techniques. Morgan Kaufmann series in data management systems.

YAN, Junchi, GONG, Min, SUN, Changhua, HUANG, Jin, CHU, Stephen M. (2012). Sales pipeline win propensity prediction: a regression approach. IBM Research. Shangai, China.

Apêndices

Apêndice A: Código fonte do classificador Árvore de Decisão

A.1 Importando as bibliotecas utilizadas

```
import pandas as pd
import numpy as np
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import RandomizedSearchCV, KFold
from scipy.stats import expon, randint, uniform
from sklearn.model_selection import cross_val_score, cross_validate
from sklearn.tree import DecisionTreeClassifier
from imblearn.pipeline import _fit_resample_one
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split as tts
from imblearn.pipeline import Pipeline, make_pipeline # doctest: +NORMALIZE_WHITESPACE
from sklearn.metrics import precision_score, recall_score, f1_score, make_scorer,
average_precision_score, precision_recall_fscore_support, accuracy_score
from sklearn import metrics
import seaborn as sn

import statistics
from sklearn.metrics import precision_recall_curve, auc
from sklearn.tree import export_graphviz
from matplotlib import pyplot as plt
from IPython.display import Image
from sklearn.externals.six import StringIO
from subprocess import check_call
import pydot
%matplotlib inline
```

A.2 Apagando as colunas com itens faltosos e colunas não utilizadas e binarizando a coluna alvo

```
df = df.dropna(axis=1, how='any')
df = df.drop(df.columns[[0, 2, 3, 4, 5, 6, 7, 8, 9]], axis=1)
df['estagio'] = df.estagio.map({'Lead':0, 'Lead Qualificado':0, 'Cliente':1})
```

A.3 Função binariza a coluna eventos com repetições

```
def main():
   # abrindo e lendo a planilha original e salvando em 'data_leads'
   # como uma lista das linhas da planilha
   leads_csv = open('Tabela_final_passo_02.csv', 'r')
```

```
data_leads = leads_csv.readlines()
leads_csv.close()

# todos os dados da planilha, com a excecao da primeira linha
# (títulos: colunas eventos e estagio)
body = data_leads[1:]
body = [line.replace('\n', '').replace('\r', '') for line in body]

# carregando dicionário com os eventos e seus índices
events = count_events(body)
# eventos desnecessários
events_block = ['members_export_51d7db9b76', 'cadastro-pro']

# inicializando as listas que representam o título
# e os dados da nova planilha a ser gerada
new_title = ['e'+str(num) for num in range(0, len(events))]
new_lines = []  # tambem poderia ser 'new_body'

# percorrendo cada linha/usuario dos dados da planilha original
for line in body:
    # pegando lista dos eventos (coluna 1)
    events_by_user = clean_event_line(line)
    # pegando o estágio (coluna 2)
    estagio = line.split(',')[-1]

    # inicializando a nova linha (binarizada) a ser escrita na nova planilha,
    # começando com a quantidade de colunas de eventos do usuario iniciadas
    # com o valor '0' mais a coluna estagio que já está pronta binarizada
    new_line = ([0] * len(events)) + [estagio]

    # percorrendo cada evento da linha/usuario por vez
    for event in events_by_user:
        # ignora eventos não desejados
        if event in events_block:
            continue
        # resgatando a posicao do evento atual na ordem em que
        # os eventos foram organizados no titulo
        position = events[event]
        # inserindo na nova linha binarizada o valor 1 para cada evento
        # encontrado de cada usuario, na posicao resgatada em 'position'
        new_line[position] += 1

    # ao percorrer todos os eventos da linha/usuario, transforma a
    # lista 'new_line' em uma string, separando os elementos por ','
    # e adicionado uma quebra de linha '\n' ao final
    new_line = [str(x) for x in new_line]
    new_line_string = ','.join(new_line) + '\n'
    # print('new_line', new_line_string)

    # adiciona a nova linha em formato de string na lista de novas
    # linhas binarizadas, ou seja, o novo 'body' da planilha
    new_lines.append(new_line_string)
```

```
    # criando e escrevendo em um novo arquivo a nova planilha, com as
    # listas 'new_title' e 'new_lines' com os valores das colunas de eventos somados pelo contador
    # e com todos os eventos existentes)
    print(events)
    new_title = new_title + ['estagio']
    new_leads_csv = open('Tabela_final_passo_03.csv', 'w')
    new_leads_csv.write(','.join(new_title)+'\n')
    new_leads_csv.writelines(new_lines)
    new_leads_csv.close()


def count_events(data):
    # inicializando dicionário de eventos
    events = {}
    # eventos ignorados
    events_block = ['members_export_51d7db9b76', 'cadastro-pro']
    # contador de eventos
    count = 0

    for line in data:
        # limpando a linha e pegando a lista dos eventos da linha/usuário
        list_events_by_user = clean_event_line(line)
        for event in list_events_by_user:
            # ignorando eventos bloqueados
            if event in events_block:
                continue
            # adicionando evento novo no dicionário
            if event not in events:
                events[event] = count
                # incrementando o contador
                count += 1
    # retorna dicionário com todos os eventos e sua contagem
    print ("\n Exibindo todos os eventos: \n")
    return events


def clean_event_line(line):
    # pegando os eventos (coluna 1)
    events_by_user = line.split(',')[:-1]
    # ignorando vírgulas entre os eventos
    events_by_user = ''.join(events_by_user)
    # retirando aspas entre os eventos
    events_by_user.replace('"', '')
    # salvando os eventos de cada linha em uma lista separados por '/'
    events_by_user = events_by_user.split('/')
    # retirando os espacos vazios do inicio e final, por
    # exemplo, ' Experimente - site ' fica 'Experimente - site'
    return [event.strip() for event in events_by_user]


if __name__ == "__main__":
```

```
main()
```

A.4 Instanciando o classificador árvore de decisão, SMOTE e pipeline e definindo o orçamento computacional

```
NUM_TRIALS = 100
smt = SMOTE()
dtree = DecisionTreeClassifier(criterion='entropy')
pipeline = Pipeline([('smt', smt), ('dtree', dtree)])
```

A.5 Criando o dicionário de hiperparâmetros

```
param_dist = {'smt__k_neighbors': randint (1, 20),
          'dtree__criterion': ["entropy"],
          'dtree__min_samples_leaf': randint (1, 10),
          'dtree__min_samples_split': randint (2, 20),
          'dtree__max_depth': randint (2, 20)}
```

A.6 Criando a pesquisa randômica com a validação cruzada aninhada, obtendo as métricas e salvando em dicionários

```
nested_recall = [0]*NUM_TRIALS
nested_precision = [0]*NUM_TRIALS
best_models = [None]*NUM_TRIALS

precisao = [0]*NUM_TRIALS
cobertura = [0]*NUM_TRIALS

matrizes = []
resultados = []
cont = []
y_real=[]
y_pred_proba=[]

f, axes = plt.subplots(1, 2, figsize=(10, 5))


for i in range(NUM_TRIALS):

   inner_cv = KFold(n_splits=5, shuffle=True, random_state=i)
   outer_cv = KFold(n_splits=5, shuffle=True, random_state=i)

   random_search = RandomizedSearchCV(pipeline,
                           param_distributions = param_dist,
                           cv = inner_cv)

   random_search.fit(X, y)
```

```
#Imprimindo as Matrizes de confusão
y_predict = random_search.predict(X)
cnf_matrix = confusion_matrix(y, y_predict)
print ("\nMatriz de confusão:",i+1)
print("\n",cnf_matrix)

#Salvando as matrizes em uma lista
matrizes.append(cnf_matrix)

#Predizendo as probabilidades para criar a curva de Precisão x Cobertura
y_predito = random_search.predict_proba(X)[:,1]
print ("\nArray de precisão e cobertura:", y_predito)

precision, recall, _ = precision_recall_curve(y, y_predito)
lab = 'Fold %d AUC=%.4f' % (i+1, auc(recall, precision))

axes[0].step(recall, precision, label=lab)
y_real.append(y)
y_pred_proba.append(y_predito)


#Imprimindo as métricas de classificação
clf_report = classification_report(y, y_predict)
print ("\nMétricas de classificação:",i+1)
print("\n",clf_report,"\n")

#Imprimindo as métricas de classificação individualmente
print ("\n\nArray de Precisão e Cobertura para avg=binary")
precisao[i] = precision_score(y, y_predict, average='binary')
print("\nPrecisão avg binary: ", precisao)

resultados.append(precisao)
resultados.append(cobertura)

cobertura[i] = recall_score(y, y_predict, average='binary')
print("\nCobertura avg binary: ", cobertura)


#Imprimimindo o melhor modelo para aquela iteração
best_models = random_search.best_params_
print ("\n\nModelo: ", best_models)
resultados.append(best_models)


#Validação cruzada para obter a Precisão
nested_precision = cross_validate(pipeline, X, y, return_train_score=False, cv=outer_cv,
                                  scoring=make_scorer(precision_score, average='binary'))
#nested_precisions[i] = nested_precision.mean()
print("\nArray de precisões para cv=outer: ", nested_precision)


#Validação cruzada para obter o recall
```

```
    nested_recall = cross_validate(pipeline, X, y, return_train_score=False, cv=outer_cv,
                            scoring=make_scorer(recall_score, average='binary'))
    #nested_recalls[i] = nested_recall.mean()
    print("\nArray de recalls para cv=outer: ", nested_recall)
    print("\n###################### FINAL DO LOOP:", i+1, "########################")
    print("\n")



y_real = np.concatenate(y_real)
y_pred_proba = np.concatenate(y_pred_proba)

precision, recall, _ = precision_recall_curve(y_real, y_pred_proba)

lab = 'Overall AUC=%.4f' % (auc(recall, precision))

axes[0].step(recall, precision, label=lab, lw=2, color='black')
axes[0].set_title('Precision-recall curve')
axes[0].set_xlabel('Recall')
axes[0].set_ylabel('Precision')
axes[0].legend(loc='lower left', fontsize='small')
```

A.7 Reportando as métricas médias

```
print ("\nMédia da precisão binária: ", statistics.mean(precisao))
print ("Média da cobertura binária: ", statistics.mean(cobertura))
print ("\n\nMatriz de confusão média:")
matrixmedia = np.mean(matrizes, axis = 0)
matrix = matrixmedia.astype(int)
print ("\n", matrix)
print ("\n#####################################################")
print ("\nDesvio padrão da precisão: ", statistics.pstdev(precisao))
print ("Desvio padrão da cobertura: ", statistics.pstdev(cobertura))
print ('Mínima precisão binária: ', min(precisao))
print ('Mínima cobertura binária: ', min(cobertura))
print ('\nMáxima precisão binária: ', max(precisao))
print ('Máxima cobertura binária: ', max(cobertura))
```

A.8 Plotando a curva de precisão e cobertura média e o gráfico de variação de precisão e cobertura

```
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.grid(True)
ax.plot(recall, precision, label=lab, lw=2, color='black')
ax.set_xlabel('Cobertura')
ax.set_ylabel('Precisão')
ax.set_title('Curva de precisão e cobertura')
```

```
#plotando a precisão e cobertura ao longo das 100 iterações
plt.figure()
plt.subplot(211)
precision_line, = plt.plot(precisao, color='g')
recall_line, = plt.plot(cobertura, color='b')
#f1_line, = plt.plot(f1_micro, color='r')
plt.ylabel("Pontuação", fontsize="14")
plt.legend([precision_line, recall_line],
        ["Precisão", "Cobertura"],
        bbox_to_anchor=(0, .4, .5, 0))
plt.title("Precisão e cobertura:",
       x=.5, y=1.1, fontsize="15")
```

A.9 Executando o modelo de maior precisão

```
smote = SMOTE(k_neighbors = 5)
clf = tree.DecisionTreeClassifier(criterion = 'entropy', min_samples_leaf=1, min_samples_split=18, max_depth=9)

pipeline_precisao = Pipeline([('smote', smote), ('clf', clf)])

pipeline_precisao.fit(X, y)
```

A.9.1 Extraindo a árvore de decisão e o ranking de importâncias do modelo em questão

```
with open("Árvore_maior_precisão.dot", "w") as f:
  f = tree.export_graphviz(clf, out_file=f)
check_call(['dot','-Tpng','Árvore_maior_precisão.dot','-o','Árvore_maior_precisão.png'])
plt.imshow(plt.imread('Árvore_maior_precisão.png'))

features = list(leads.columns[0:46])
importances = zip(clf.feature_importances_, features)
indices = np.argsort(importances)
print("Feature ranking:\n")
for importance, feature in sorted(importances, reverse=True):
  print("%s: %f%%" % (feature, importance))
```

A.10 Executando o modelo de maior cobertura

```
smote_cobertura = SMOTE(k_neighbors = 4)
clf_cobertura = tree.DecisionTreeClassifier(criterion = 'entropy', min_samples_leaf=4, min_samples_split=11, max_depth=5)

pipeline_cobertura = Pipeline([('smote_cobertura', smote_cobertura), ('clf_cobertura', clf_cobertura)])

pipeline_cobertura.fit(X, y)
```

A.10.1 Extraindo a árvore de decisão e o ranking de importâncias deste modelo

```
with open("Árvore_maior_cobertura.dot", "w") as f:
   f = tree.export_graphviz(clf_cobertura, out_file=f)
check_call(['dot','-Tpng','Árvore_maior_cobertura.dot','-o','Árvore_maior_cobertura.png'])
plt.imshow(plt.imread('Árvore_maior_cobertura.png'))

importances_cobertura = zip(clf_cobertura.feature_importances_, features)
indices = np.argsort(importances_cobertura)
print("Feature ranking:\n")
for importance, feature in sorted(importances_cobertura, reverse=True):
   print("%s: %f%%" % (feature, importance))
```

Apêndice B: Código fonte do classificador Naive Bayes

B.1 Importando as bibliotecas utilizadas neste classificador

```
import pandas as pd
import numpy as np
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import RandomizedSearchCV, KFold
from scipy.stats import expon, randint, uniform
from sklearn.model_selection import cross_val_score, cross_validate
from sklearn.tree import DecisionTreeClassifier
from imblearn.pipeline import _fit_resample_one
from sklearn.naive_bayes import BernoulliNB
from sklearn.tree import export_graphviz
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split as tts
from imblearn.pipeline import Pipeline, make_pipeline # doctest: +NORMALIZE_WHITESPACE
from matplotlib import pyplot as plt
%matplotlib inline
from sklearn.metrics import precision_score, recall_score, f1_score, make_scorer,
average_precision_score, precision_recall_fscore_support, accuracy_score
from sklearn import metrics
import seaborn as sn
import statistics
from sklearn.metrics import roc_curve
from sklearn.metrics import precision_recall_curve, auc
from sklearn.metrics import roc_auc_score

from sklearn.tree import export_graphviz
from matplotlib import pyplot as plt
from IPython.display import Image
from sklearn.externals.six import StringIO
```

```
from subprocess import check_call
import pydot
```

B.2 Apagando as colunas com itens faltosos e colunas não utilizadas e binarizando a coluna alvo

```
df = df.dropna(axis=1, how='any')
df = df.drop(df.columns[[0, 2, 3, 4, 5, 6, 7, 8, 9]], axis=1)
df['estagio'] = df.estagio.map({'Lead':0, 'Lead Qualificado':0, 'Cliente':1})
```

B.3 Função binariza a coluna eventos com repetições

```
def main():
    # abrindo e lendo a planilha original e salvando em 'data_leads'
    # como uma lista das linhas da planilha
    leads_csv = open('Tabela_final_passo_02.csv', 'r')
    data_leads = leads_csv.readlines()
    leads_csv.close()

    # todos os dados da planilha, com a excecao da primeira linha
    # (títulos: colunas eventos e estagio)
    body = data_leads[1:]
    body = [line.replace('\n', '').replace('\r', '') for line in body]

    # carregando dicionário com os eventos e seus índices
    events = count_events(body)
    # eventos desnecessários
    events_block = ['members_export_51d7db9b76', 'cadastro-pro']

    # inicializando as listas que representam o título
    # e os dados da nova planilha a ser gerada
    new_title = ['e'+str(num) for num in range(0, len(events))]
    new_lines = []  # tambem poderia ser 'new_body'

    # percorrendo cada linha/usuario dos dados da planilha original
    for line in body:
        # pegando lista dos eventos (coluna 1)
        events_by_user = clean_event_line(line)
        # pegando o estágio (coluna 2)
        estagio = line.split(',')[-1]

        # inicializando a nova linha (binarizada) a ser escrita na nova planilha,
        # começando com a quantidade de colunas de eventos do usuario iniciadas
        # com o valor '0' mais a coluna estagio que já está pronta binarizada
        new_line = ([0] * len(events)) + [estagio]

        # percorrendo cada evento da linha/usuario por vez
        for event in events_by_user:
            # ignora eventos não desejados
            if event in events_block:
```

```
            continue
        # resgatando a posicao do evento atual na ordem em que
        # os eventos foram organizados no titulo
        position = events[event]
        # inserindo na nova linha binarizada o valor 1 para cada evento
        # encontrado de cada usuario, na posicao resgatada em 'position'
        new_line[position] += 1

    # ao percorrer todos os eventos da linha/usuario, transforma a
    # lista 'new_line' em uma string, separando os elementos por ','
    # e adicionado uma quebra de linha '\n' ao final
    new_line = [str(x) for x in new_line]
    new_line_string = ','.join(new_line) + '\n'
    # print('new_line', new_line_string)

    # adiciona a nova linha em formato de string na lista de novas
    # linhas binarizadas, ou seja, o novo 'body' da planilha
    new_lines.append(new_line_string)

# criando e escrevendo em um novo arquivo a nova planilha, com as
# listas 'new_title' e 'new_lines' com os valores das colunas de eventos somados pelo contador
# e com todos os eventos existentes)
print(events)
new_title = new_title + ['estagio']
new_leads_csv = open('Tabela_final_passo_03.csv', 'w')
new_leads_csv.write(','.join(new_title)+'\n')
new_leads_csv.writelines(new_lines)
new_leads_csv.close()


def count_events(data):
    # inicializando dicionário de eventos
    events = {}
    # eventos ignorados
    events_block = ['members_export_51d7db9b76', 'cadastro-pro']
    # contador de eventos
    count = 0

    for line in data:
        # limpando a linha e pegando a lista dos eventos da linha/usuário
        list_events_by_user = clean_event_line(line)
        for event in list_events_by_user:
            # ignorando eventos bloqueados
            if event in events_block:
                continue
            # adicionando evento novo no dicionário
            if event not in events:
                events[event] = count
                # incrementando o contador
                count += 1
    # retorna dicionário com todos os eventos e sua contagem
    print ("\n Exibindo todos os eventos: \n")
```

```
    return events


def clean_event_line(line):
    # pegando os eventos (coluna 1)
    events_by_user = line.split(',')[:-1]
    # ignorando vírgulas entre os eventos
    events_by_user = ''.join(events_by_user)
    # retirando aspas entre os eventos
    events_by_user.replace('"', '')
    # salvando os eventos de cada linha em uma lista separados por '/'
    events_by_user = events_by_user.split('/')
    # retirando os espacos vazios do inicio e final, por
    # exemplo, ' Experimente - site ' fica 'Experimente - site'
    return [event.strip() for event in events_by_user]


if __name__ == "__main__":
    main()
```

B.4 Instanciando o classificador Naive Bayes, SMOTE e pipeline e definindo o orçamento computacional

```
NUM_TRIALS = 100
smt = SMOTE()
nb = BernoulliNB()
pipeline = Pipeline([('smt', smt), ('nb', nb)])
```

B.5 Criando o dicionário de hiperparâmetros

```
param_dist = {'smt__k_neighbors': randint (1, 20)}
```

B.6 Criando a pesquisa randômica com a validação cruzada aninhada, obtendo as métricas e salvando em dicionários

```
nested_recall = [0]*NUM_TRIALS
nested_precision = [0]*NUM_TRIALS
best_models = [None]*NUM_TRIALS

precisao = [0]*NUM_TRIALS
cobertura = [0]*NUM_TRIALS

matrizes = []
resultados = []
cont = []
y_real=[]
y_pred_proba=[]
```

```
f, axes = plt.subplots(1, 2, figsize=(10, 5))

for i in range(NUM_TRIALS):

   inner_cv = KFold(n_splits=5, shuffle=True, random_state=i)
   outer_cv = KFold(n_splits=5, shuffle=True, random_state=i)

   random_search = RandomizedSearchCV(pipeline,
                           param_distributions = param_dist,
                           cv = inner_cv)

   random_search.fit(X, y)

   #Imprimindo as Matrizes de confusão
   y_predict = random_search.predict(X)
   cnf_matrix = confusion_matrix(y, y_predict)
   print ("\nMatriz de confusão:",i+1)
   print("\n",cnf_matrix)

   #Salvando as matrizes em uma lista
   matrizes.append(confusion_matrix(y, y_predict))


   #Predizendo as probabilidades para criar a curva de Precisão x Cobertura
   y_predito = random_search.predict_proba(X)[:,1]
   print ("\nArray de precisão e cobertura:", y_predito)

   precision, recall, _ = precision_recall_curve(y, y_predito)
   lab = 'Fold %d AUC=%.4f' % (i+1, auc(recall, precision))

   axes[0].step(recall, precision, label=lab)
   y_real.append(y)
   y_pred_proba.append(y_predito)


   #Imprimindo as métricas de classificação
   clf_report = classification_report(y, y_predict)
   print ("\nMétricas de classificação:",i+1)
   print("\n",clf_report,"\n")

   #Imprimindo as métricas de classificação individualmente
   print ("\n\nArray de Precisão e Cobertura para avg=binary")
   precisao[i] = precision_score(y, y_predict, average='binary')
   print("\nPrecisão avg binary: ", precisao)

   resultados.append(precisao)
   resultados.append(cobertura)

   cobertura[i] = recall_score(y, y_predict, average='binary')
   print("\nCobertura avg binary: ", cobertura)
```

```
    #Imprimimindo o melhor modelo para aquela iteração
    best_models = random_search.best_params_
    print ("\n\nModelo: ", best_models)
    resultados.append(best_models)


    #Validação cruzada para obter a Precisão
    nested_precision = cross_validate(pipeline, X, y, return_train_score=False, cv=outer_cv,
                                scoring=make_scorer(precision_score, average='binary'))
    #nested_precisions[i] = nested_precision.mean()
    print("\nArray de precisões para cv=outer: ", nested_precision)


    #Validação cruzada para obter o recall
    nested_recall = cross_validate(pipeline, X, y, return_train_score=False, cv=outer_cv,
                            scoring=make_scorer(recall_score, average='binary'))
    #nested_recalls[i] = nested_recall.mean()
    print("\nArray de recalls para cv=outer: ", nested_recall)
    print("\n###################### FINAL DO LOOP:", i+1, "########################")
    print("\n")

y_real = np.concatenate(y_real)
y_pred_proba = np.concatenate(y_pred_proba)

precision, recall, _ = precision_recall_curve(y_real, y_pred_proba)

lab = 'Overall AUC=%.4f' % (auc(recall, precision))

axes[0].step(recall, precision, label=lab, lw=2, color='black')
axes[0].set_title('Precision-recall curve')
axes[0].set_xlabel('Recall')
axes[0].set_ylabel('Precision')
axes[0].legend(loc='lower left', fontsize='small')
```

B.7 Reportando as métricas médias

```
print ("\nMédia da precisão binária: ", statistics.mean(precisao))
print ("Média da cobertura binária: ", statistics.mean(cobertura))
print ("\n\nMatriz de confusão média:")
matrixmedia = np.mean(matrizes, axis = 0)
matrix = matrixmedia.astype(int)
print ("\n", matrix)
print ("\n#####################################################")
print ("\nDesvio padrão da precisão: ", statistics.pstdev(precisao))
print ("Desvio padrão da cobertura: ", statistics.pstdev(cobertura))
print ('Mínima precisão binária: ', min(precisao))
print ('Mínima cobertura binária: ', min(cobertura))
print ('\nMáxima precisão binária: ', max(precisao))
print ('Máxima cobertura binária: ', max(cobertura))
```

B.8 Plotando a curva de precisão e cobertura média e o gráfico de variação de precisão e cobertura

```
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
#fig, ax = plt.subplots(figsize=(8,8))
ax.grid(True)
ax.plot(recall, precision, label=lab, lw=2, color='black')
ax.set_xlabel('Cobertura')
ax.set_ylabel('Precisão')
ax.set_title('Curva de precisão e cobertura')

#plotando a precisão e cobertura ao longo das 100 iterações
plt.figure()
plt.subplot(211)
precision_line, = plt.plot(precisao, color='g')
recall_line, = plt.plot(cobertura, color='b')
#f1_line, = plt.plot(f1_micro, color='r')
plt.ylabel("score", fontsize="14")
plt.legend([precision_line, recall_line],
        ["Precisão", "Cobertura"],
        bbox_to_anchor=(0, .4, .5, 0))
plt.title("Gráfico de precisão e cobertura",
       x=.5, y=1.1, fontsize="15")
```