Instituto de MATEMÁTICA E ESTATÍSTICA UFRGS

DEPARTAMENTO DE ESTATÍSTICA
E-mail: dest@mat.ufrgs.br

Trabalho de Conclusão de Curso

Classificação de texto de reclamações de empreendimentos imobiliários utilizando *Machine Learning*

Bernardo Altenbernd

3 de junho de 2021

Bernardo Altenbernd

Classificação de texto de reclamações de empreendimentos imobiliários utilizando *Machine Learning*

Trabalho de Conclusão apresentado à comissão de Graduação do Departamento de Estatística da Universidade Federal do Rio Grande do Sul, como parte dos requisitos para obtenção do título de Bacharel em Estatística.

Orientadora: Profa. Dra. Márcia Helena Barbian
Co-Orientadora: Profa. Dra. Márcia Elisa Soares Echeveste

Porto Alegre
Maio de 2021

Bernardo Altenbernd

Classificação de texto de reclamações de empreendimentos imobiliários utilizando *Machine Learning*

Este Trabalho foi julgado adequado para obtenção dos créditos da disciplina Trabalho de Conclusão de Curso em Estatística e aprovado em sua forma final pela Orientadora e pela Banca Examinadora.

Orientadora:_____
Profa. Dra. Márcia Helena Barbian, UFMG
Doutora pela Universidade Federal de Minas Gerais, Belo Horizonte, MG
Co-Orientadora:_____
Profa. Dra. Márcia Elisa Soares Echeveste, UFRGS
Doutora pela Universidade Federal do Rio Grande do Sul, Porto Alegre, RS

Banca Examinadora:

Prof. Dr. João Henrique Ferreira Flores, UFRGS
Doutor pela Universidade Federal do Rio Grande do Sul, Porto Alegre, RS

Porto Alegre
Maio de 2021

"Não há acaso, sina, destino, que possa limitar, impedir ou controlar a firme resolução de uma alma determinada." (Ella Wheller Wilcox)

Agradecimentos

À minha mãe Leila pela inspiração de vida e apoio a todas as minhas decisões.

À minha irmã Bibiana, pelo exemplo a ser seguido, ajuda e reconhecimento.

À minha avó Therezinha pela base educacional e disciplina.

Ao meu pai André pelos exemplos de ética e honestidade.

À minha noiva Caroline pela compreensão e parceria.

Aos meus irmãos Edgar e Arthur, além de todos os familiares e amigos pela participação e torcida pela minha caminhada.

Às minhas orientadoras pela paciência, confiança e apoio.

Aos professores do curso de estatística pelos ensinamentos.

À UFRGS pela oportunidade de ensino.

Resumo

As empresas construtoras de empreendimentos imobiliários ao produzirem os imóveis para venda, estão sujeitas a cometerem falhas construtivas, sejam estas por problemas em processos produtivos ou nos materiais utilizados. Embora existam mecanismos de melhoria na qualidade das habitações, defeitos seguem ocorrendo e demandando ações corretivas em um volume considerável a ponto das empresas estruturarem setores de assistência técnica para melhor lidar e atender as reclamações. O presente trabalho propõe métodos automáticos de classificação de texto que recebem os registros textuais e auxiliam na categorização técnica das reclamações, necessária para dar prosseguimento ao processo de reparo. Metodologias eficazes de identificação podem auxiliar os departamentos de gestão com agilidade e assertividade nas ações, ou ainda gerar *insights* de negócio em favor de diferentes práticas que reduzam custos e otimizem ganhos. Nos dias atuais os avanços tecnológicos viabilizam o tratamento de dados não estruturados de muitas formas diferentes, sendo que foram selecionados alguns dos métodos de *machine learning* mais utilizados na literatura para este fim que são *Naive Bayes*, *Support Vector Machines*, *Random Forest* e *Boosting*.

Palavras-Chave: Classificação de Texto, Reclamações, *Machine Learning*, *Naive Bayes*, *Support Vector Machines*, *Random Forest*, *Boosting*.

Abstract

The construction companies that build big projects when producing the units for sale, are subject to commit construction failures, whether these are problems in production processes or in the materials used. Although mechanisms for improving the quality of housing exist, defects continue to occur and demand corrective actions in a considerable volume to the point that companies structure technical assistance sectors to better deal with and respond to complaints. This work proposes automatic text classification methods that receive textual records and assist in the necessary technical categorization to proceed with the repair process. Effective identification methodologies can assist management departments with agility and assertiveness in actions, or even generate business insights in favor of different practices that reduce costs and optimize gains. Nowadays technological advances make it possible to treat unstructured data in many different ways, with some of the most used machine learning methods used in the literature for this purpose being Naive Bayes, Support Vector Machines, Random Forest, Bossting.

Keywords: Text Classification, Complaints, Machine Learning, Naive Bayes, Support Vector Machines, Random Forest, Boosting.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

A área da construção civil vem sendo submetida a uma exigência de qualidade de seus clientes cada vez mais criteriosa com relação aos defeitos em seus imóveis (Bazzan, 2019). Problemas construtivos muitas vezes não são identificados no momento da entrega do imóvel, mas sim com o passar dos dias de uso, ou ainda podem surgir com um tempo menor do que o esperado, dentro do período de garantia oferecido pelas construtoras. As ocorrências reportadas pelos consumidores podem alimentar as bases que viabilizam análises que serão importantes para grande economia de recursos ao evitar defeitos recorrentes, bem como geração de *insights* de negócio como indicadores para fornecedores ou materiais mais adequados. Segundo Cupertino e Brandstetter (2015), dados de assistência técnica embasam a tomada de decisões para promover ações que evitem reincidências dos problemas mais frequentes nos próximos empreendimentos.

Este estudo faz parte de uma pesquisa aplicada ao setor de assistência técnica de uma construtora de grande porte localizada no Sul do Brasil, para a gestão dos registros de reclamações dos clientes, neste caso moradores dos empreendimentos entregues pela empresa. O conjunto de reclamações é composto por uma grande quantidade de registros textuais em linguagem natural e geram uma massa de dados registrada em diferentes formas de expressão. As reclamações são relativas à problemas construtivos como fissuras nas paredes, descolamento de revestimento cerâmico, infiltração, entre outras falhas. Os dados disponíveis são reais e representam 2741 reclamações realizadas entre janeiro de 2017 a março de 2018.

A natureza dos dados é de livre expressão de muitas pessoas diferentes, o que implica na dificuldade de interpretação imediata do seu conteúdo. Além disso cada falha construtiva demanda algum tipo de classificação técnica específica (Das e Chew, 2011), para que seja encaminhada ao setor correto para sua solução. Essa classificação transcende o conhecimento dos clientes, os quais desconhecem termos técnicos no âmbito da construção civil. Fato este que força uma ação de intervenção para categorizar previamente cada reclamação. Outros problemas são que a falta de uniformização na linguagem pode acarretar em erros de entendimento, uma vez que ações de reparos são tomadas de acordo com a percepção do supervisor de cada setor, bem como as possíveis causas ou origens do problema não são registradas, perdendo-se rastreabilidade de informação (Bazzan, 2019). Em outras palavras, o registro sobre as origens do problema acabam sendo extintas no momento que ele é resolvido. Portanto os registros de reclamações são utilizados basicamente para processos corretivos, na qual o ideal é a adoção de processos de melhoria com o intuito de tornar a prática preventiva. Somando a isso, observa-se a ausência de

procedimentos de análises estatísticas para esse tipo de dados, o que sinaliza um entendimento enviesado do cenário de problemas de qualidade por parte dos setores de gestão.

Neste trabalho são apresentados métodos de estatística descritiva, que são técnicas destinadas a resumir e descrever os dados a fim de extrair informações de interesse (Magalhães e de Lima, 2004), e técnicas estatísticas de classificação de texto, que segundo Arif-Uz-Zaman et al. (2017) são algoritmos que têm como finalidade classificar automaticamente um conjunto de registros textuais em categorias predefinidas. Tendo em vista que o banco de dados conta com as classificações técnicas atribuídas às reclamações, as quais foram feitas manualmente com o propósito de estudar suas relações. Com isso, é atendida a suposição para a utilização de *Machine Learning* supervisionado para classificação automatizada das reclamações.

Para este estudo propõe-se inicialmente a adoção da representação do documento de registros na abordagem *bag-of-words*, ou seja, a ordem das posições das palavras não é levada em consideração, apenas a sua frequência no documento todo (Jurasky e Martin, 2000). Para as análises descritivas serão gerados objetos como nuvem de palavras, tabela de frequências e gráfico de redes (Gunay et al., 2019). Os ajustes dos modelos de classificação se darão por meio dos métodos *Naive Bayes*, *Random Forest*, *Boosting* e *Support Vector Machine* (SVM). As reclamações serão classificadas como uma das seis categorias da variável *Sistema*: Sistemas prediais, Esquadrias, Vedações horizontais, Vedações verticais, Estrutura e Complementares. As qualificações contidas em *Sistema* corresponde ao primeiro nível de detalhamento das falhas reportadas, segundo a classificação proposta por Bazzan (2019).

Na literatura são encontradas diferentes técnicas para classificação de texto baseados em *statistical learning*, como *Naive Bayes* (Lantz, 2013), Árvores de Decisão (Chi et al., 2012), *Random Forest* (de Castro, 2017), *Boosting* (Santos et al., 2019), *Support Vector Machines* (SVM) (de Pádua Moreira e Nascimento, 2012; Chi et al., 2014), *k-nearest neighbour* (kNN)(Wang et al., 2014) e Redes Neurais Artificiais (Basu et al., 2003). É comum que pesquisadores façam o uso de mais de uma técnica para fins de comparação, como é o caso dos estudos de Joachims (1998) que obteve melhores resultados com SVM em uma grande base de dados se comparado com *Naive Bayes* e kNN (Lantz, 2013; Wang et al., 2014). Seguindo a prática de pesquisas anteriores, este trabalho apresentará uma análise descritiva dos dados em conjunto com uma comparação de quatro diferentes classificadores de texto conhecidos: *Naive Bayes*, *Random Forest*, *Boosting* e SVM.

O objetivo geral deste trabalho é classificar de forma automática as reclamações dos clientes, por meio de técnicas de *Machine Learning*, e associá-las com falhas de projeto que auxiliem no entendimento de problemas de qualidade. Para isso, serão feitas análises descritivas textuais para ajudar na identificação dos problemas mais reportados no conjunto de reclamações, serão ajustados modelos de classificação para as reclamações entrantes de acordo com o nível de caracterização *Sistemas*. Para a leitura e manipulação do banco de dados, bem como para modelagem e comparações, será usado o software *R* versão 4.0.4 (R Core Team, 2021).

2 *Statistical Learning*

2.1 *Statistical Learning*

Statistical Learning é uma ciência recentemente desenvolvida na área da estatística em paralelo com a ciência da computação, e consiste em um conjunto de ferramentas para modelagem de dados complexos utilizando métodos clássicos de estatística, bem como metodologias mais modernas (James et al., 2013). Essas ferramentas podem ser classificadas como aprendizagem supervisionada ou não supervisionada. Aprendizagem supervisionada é ajustar um modelo estatístico para prever ou predizer um resultado baseado em um ou mais dados de entrada com classificações prévias, ou seja, para cada observação x_i, $i = 1, \ldots, n$ há uma resposta y_i associada. Já na aprendizagem não supervisionada observamos igualmente o vetor x_i , $i = 1, \ldots, n$ porém, com a ausência da variável resposta associada para supervisão, neste caso os algoritmos objetivam entender as relações entre variáveis e observações. Para o banco de dados utilizado neste trabalho, as reclamações estão classificadas previamente para a variável categórica *Sistema*, portanto serão abordadas apenas técnicas de classificação e de aprendizado supervisionado.

Supomos uma variável resposta Y e p diferentes variáveis explicativas, $X_1, X_2, \ldots, X_p$, assume-se que há uma relação entre Y e $\mathbf{X} = X_1, X_2, \ldots, X_p$ que pode ser escrita como

$$Y = f(\mathbf{X}) + \varepsilon, \tag{2.1}$$

onde ε é o erro aleatório independente de $\mathbf{X}$ e com média zero, e f é uma função desconhecida de $\mathbf{X}$ que representa informações sistemáticas que $\mathbf{X}$ fornece sobre Y. Em sua essência, *statistical learning* é um conjunto de abordagens e técnicas de estimar f com o objetivo de fazer predições ou previsões sobre Y (James et al., 2013). Dispomos de um conjunto de n observações, que são chamadas de dados de treinamento porque serão usadas para treinar ou ensinar o método escolhido de como estimar f. Em outras palavras, queremos encontrar uma função $\hat{f}$ onde $Y \approx \hat{f}(\mathbf{X})$ para cada observação $(\mathbf{X}, Y)$.

Dentre os diversos métodos de treinamento estudados em *statistical learning*, alguns são mais restritivos, do ponto de vista que contemplam poucas formas de estimar f. Por exemplo, regressão linear é uma abordagem relativamente inflexível, pois pode produzir apenas funções lineares, representadas por retas ou planos. Por outro lado, outros métodos são considerados mais flexíveis, como *Boosting* ou SVM, que podem fornecer uma gama maior de formas possíveis de estimar f.

Os hiperparâmetros são características ajustáveis, próprias de cada modelo, e permitem controlar o processo de treinamento dos métodos. O processo de refinamento destes valores em busca dos melhores resultados é chamado de *tuning*.

2.1.1 Variância

A variância em métodos de *statistical learning* é o valor pelo qual $\hat{f}$ mudaria ao fazer o uso de uma diferente base de treinamento. Uma vez que um conjunto de dados é usado para ajuste do modelo, diferentes bancos de treino resultarão em diferentes $\hat{f}$, mas o cenário ideal é que a estimação de f não tenha muita variação para diferentes dados de treino. Se um método possui alta variância, pequenas alterações no banco podem resultar em grandes mudanças em $\hat{f}$, métodos de *statistical learning* mais flexíveis tendem a ter maiores variâncias (James et al., 2013). Modelos com alta variância captam características excessivamente específicas do banco de dados analisado, são ruídos aleatórios que não representam o sinal da população alvo da análise.

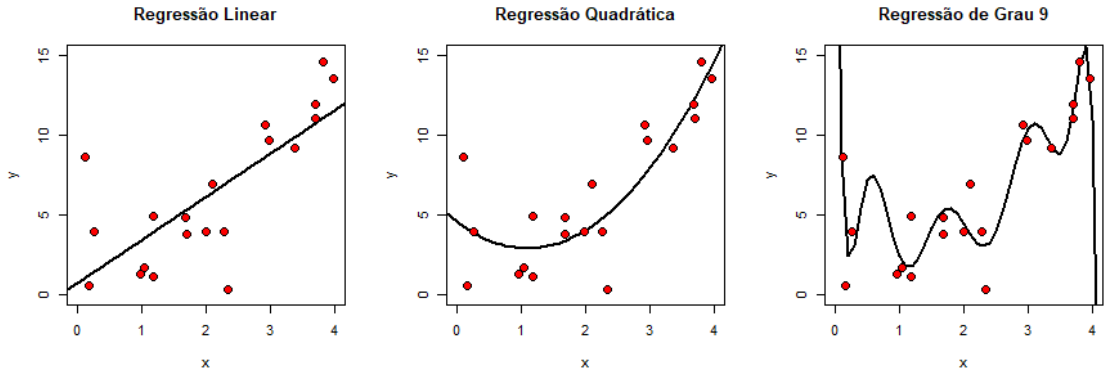
2.1.2 Vício

Também conhecido como o viés dos modelos de *statistical learning*, o vício refere-se ao erro que é introduzido com a representação de um problema real complexo, por um modelo mais simples. Como exemplo, uma regressão linear assume uma relação linear entre Y e $X_1, X_2, \ldots, X_p$, mas para alguns problemas dificilmente esta relação seja tão trivial, neste caso a regressão linear resultará em um viés na estimativa de f. Neste contexto, se a verdadeira f oferecer uma relação não linear, independentemente do tamanho amostral, não será possível produzir uma estimativa acurada de f usando simplesmente uma regressão linear. Isto aconteceria em virtude de que um modelo menos flexível resulta em um viés maior (James et al., 2013).

De um modo geral, devemos levar em consideração o equilíbrio entre variância e viés na escolha de um modelo de *statistical learning*, já que modelos mais flexíveis resultam no decréscimo no viés mas um acréscimo na variância.

Suponha um conjunto de dados ajustado por três modelos, regressão linear, regressão quadrática e regressão de grau 9. Através da Figura 2.1 observa-se que a regressão linear é um modelo muito simples, com viés elevado por não identificar um comportamento aparentemente não linear dos dados. A regressão de grau 9 aparenta ser um modelo muito complexo, elevando a variância pela sua alta flexibilidade. Já a regressão quadrática, é um exemplo mais equilibrado na relação viés e variância para modelagem dos dados. Em uma situação hipotética de retirada de uma nova amostra da mesma distribuição do exemplo, a função obtida e representada pelo terceiro gráfico certamente não estaria bem ajustada aos novos dados. Este problema acontece pela explicação muito específica do modelo para a primeira amostra, e é denominado de *overfitting*.

Figura 2.1: Relação Viés e Variância.



2.2 Validação Cruzada

Segundo James et al. (2013), métodos de reamostragem são ferramentas indispensáveis para a estatística moderna. Essas técnicas envolvem repetidas extrações de amostras de uma base de treinamento seguido de reajustes do modelo de interesse para cada amostra. As abordagens de reamostragem podem ter alto custo computacional, uma vez que se trata da repetição de um mesmo método estatístico usando diferentes subconjuntos de dados diferentes do banco de treinamento. No entanto, as exigências computacionais desses métodos podem ser atendidas com maior facilidade graças aos avanços tecnológicos recentes.

O método de validação cruzada chamado de *k-Fold* consiste na divisão do conjunto de observações em k grupos de aproximadamente mesmo tamanho. O primeiro grupo será tratado como conjunto de validação, enquanto que os demais grupos serão usados para ajuste. O erro quadrático médio (EQM) será calculado de acordo com as observações do grupo em separado. Este procedimento é repetido k vezes, e a cada vez um diferente grupo irá servir para validação. Este processo resultará em k estimativas de erros dos testes, $EQM_1, EQM_2, \ldots, EQM_k$. E a estimativa *k-fold cross-validation* será obtida com a média destes valores,

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} EQM_i. \tag{2.2}$$

Esta é uma prática para se obter informações adicionais sobre a capacidade de generalização do modelo ajustado, o objetivo é determinar o que se espera de como o procedimento de *statistical learning* desempenhará as classificações diante do banco de teste (James et al., 2013). A validação cruzada também ajuda na escolha de hiperparâmetros dos métodos e na avaliação da relação vício e variância. Para este trabalho, a base de treinamento foi separada em cinco partes para aplicação da validação cruzada.

2.3 Classificação de Texto

Os modelos de regressão linear assumem que uma variável resposta Y é quantitativa, mas em muitos casos a variável resposta é qualitativa. Um exemplo que pode

ser citado é a variável cor dos olhos, podendo ser referida também como uma variável categórica, que pode assumir os valores verde, azul ou castanho. Os processos que objetivam predizer respostas qualitativas são conhecidos como *classificação*.

Segundo Wang et al. (2017) classificação trata-se de uma das tarefas mais importantes de *Machine Learning*: o problema de identificar qual conjunto ou categoria uma nova observação pertence. Como neste trabalho as observações com as classes já atribuídas, a classificação dos novos registros se dará com o uso de classificadores de aprendizagem supervisionada. Dentre os vários algoritmos de aprendizagem, foram selecionados quatro dos mais utilizados na literatura: *Naive Bayes*, *Random Forest*, *Support Vector Machine* e *Boosting*.

No contexto deste trabalho, dos Santos (2015) conceitua a tarefa de classificação textual como tornar automático o processo de organização de registros textuais em um conjunto de categorias previamente definidas. No âmbito de *statistical learning*, ainda conforme dos Santos (2015) a classificação de texto passa por quatro etapas. Primeiramente é feita a classificação manual dos dados, é quando alguém com domínio sobre o assunto lê e classifica um conjunto de documentos, por conta desta etapa que os métodos são considerados supervisionados. O próximo passo é o pré-processamento do texto, que consiste em transformar os dados em uma representação que viabilize a aplicação dos métodos de classificação, ou seja, tornar os documentos de texto compreensíveis computacionalmente. Em seguida são aplicadas regras de classificação, e por fim é feita a verificação da qualidade da classificação através de medidas de performance e análise da matriz de confusão (dos Santos, 2015).

2.4 Análise de *Keyness*

O termo *Keyness*, remetido de "palavra-chave"em inglês, é definido como uma palavra que ocorre com frequência incomum em um determinado texto em comparação com um *corpus* de referência. O objetivo desta análise é estabelecer palavras em um texto que forneçam a representação de conceitos importantes para o conjunto textual (Gabrielatos, 2018).

A importância de palavras para um *corpus* pode ser calculada através da comparação da sua frequência em toda a base de dados com um conjunto de referência. Em outras palavras, é feita a avaliação se um determinado termo ocorre mais vezes em um subconjunto, uma determinada classe por exemplo, do que no conjunto total de registros. Pode-se quantificar a atração relativa de cada palavra para uma classe alvo, através de métricas calculadas por verossimilhança ou teste Qui-Quadrado.

2.5 Classificador *Naive Bayes*

É um conjunto de algoritmos de *Machine Learning* supervisionado que aplicam o Teorema de Bayes em conjunto com a pressuposição de que, dada uma determinada categoria, há independência condicional entre os atributos (dos Santos, 2015).

Considere que o banco de dados com opiniões de clientes, denotado por R_i, sendo $i = 1, \ldots, n$ registros, para cada uma dessas observações é atribuída uma classe c_j onde $j = 1, \ldots, a$ classes. Classificador ingênuo de *Bayes* é um método probabilístico, então para cada dado entrante o algoritmo vai retornar uma classe estimada $\hat{c}$ que será a de maior probabilidade a *posteriori*:

$$\hat{c} = \max\{P(c_j|R_i)\}. \tag{2.3}$$

Se incluirmos o Teorema de Bayes em (2.3):

$$\hat{c} = \max\{P(c_j|R_i)\} = \max\left\{\frac{P(R_i|c_j)P(c_j)}{P(R_i)}\right\}. \tag{2.4}$$

É possível simplificar a equação (2.4) de três formas:

- Desconsiderar o denominador $P(R_i)$, isso é possível pois este valor não muda para todas as classes, uma vez que os cálculos são feitos todos a partir do mesmo documento R_i;
- Considerar, pela ideia de *bag-of-words*, que as posições das palavras não importam, então representamos o documento $\mathbf{R}$ como o conjunto de palavras x_k, $k = 1, \ldots, m$ e m o número total de palavras no banco, e que são ponderadas apenas pela sua identidade;
- Pela chamada suposição ingênua de Bayes, que considera que as probabilidades $P(x_k|c_j)$, são independentes e portanto podem ser multiplicadas.

Neste caso a equação final para o classificador *Naive Bayes* é:

$$\hat{c} = \max\left\{P(c_j)\prod_{k=1}^{m} P(x_k|c_j)\right\}. \tag{2.5}$$

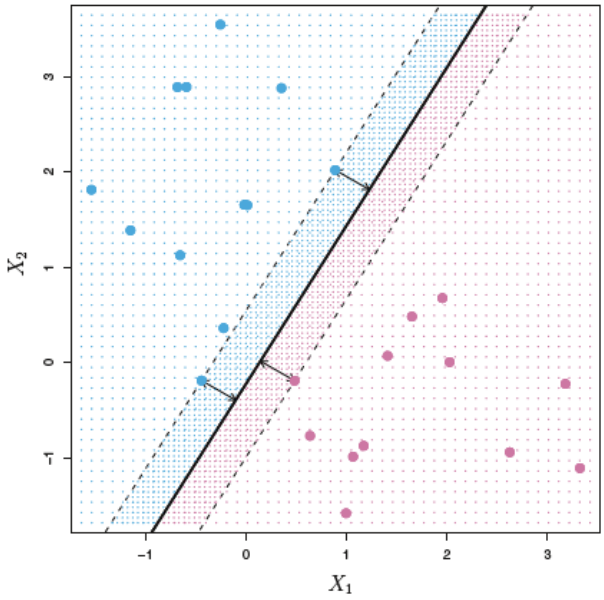
Esta é uma técnica que se mantém popular ao longo dos anos para classificação de documentos de texto. Conforme Jurasky e Martin (2000) o classificador *Naive Bayes* é assim chamado por assumir de forma simplista sobre a dependência entre as covariáveis, mas apesar de que a suposição de independência possa parecer muito otimista, esses classificadores muitas vezes superam em resultados alternativas mais sofisticadas (Hastie et al., 2009).

2.6 *Support Vector Machines*

Segundo dos Santos (2015) as máquinas de vetores de suporte, tradução do inglês de *Support Vector Machines* (SVM), possuem potencial para classificação textual por trabalharem com grandes espaços de atributos. A sua utilização é motivada com regras de decisão a partir da construção de hiperplanos em espaços vetoriais que permitam classificar bem os dados. Um hiperplano é um subespaço vetorial de dimensão $d-1$ de um espaço vetorial d-dimensional, então uma reta por exemplo é um hiperplano de um plano.

Na imagem abaixo o objetivo é determinar a qual de duas características um conjunto de objetos pertence. Representadas por círculos rosas ou azuis, ao plotarmos estes objetos em um plano, a reta traçada de tal forma a dividir os dados em dois grupos é um hiperplano.

Figura 2.2: Classificação linear com um hiperplano.



Fonte: James et al. (2013)

Observa-se que para classificar bem ambas as classes o ideal é que o hiperplano esteja o mais distante possível de forma simultânea dos dados. Os objetos postados próximos à fronteira de decisão são os usados para determinar a regra ótima de classificação, os quais são chamados de vetores de suporte (dos Santos, 2015). Ainda segundo dos Santos (2015), o fato de que problemas de classificação de texto são em sua maioria linearmente separáveis sugere o uso de SVM linear, que tem como característica maior simplicidade teórica e com menor tempo de processamento computacional se comparado com o SVM não linear.

Para o desenvolvimento formal da técnica, levemos em conta a sua versão mais simples, o SVM linear com dados linearmente separáveis, chamado também de problema com margem rígida. Suponha que para cada categoria c_j de um banco de treino T esteja associado a um vetor de características x_k no espaço de dados X. Um classificador linear pode ser representado pela equação de um hiperplano

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = 0 \tag{2.6}$$

onde o produto interno entre os vetores $\mathbf{w}$ e $\mathbf{x}$ é o vetor normal ao hiperplano, e b é o viés, de modo que $\frac{b}{||\mathbf{w}||}$ é a distância do hiperplano em relação à origem.

A função divide o espaço dos dados X em duas regiões: $\mathbf{w} \cdot \mathbf{x} + b > 0$ e $\mathbf{w} \cdot \mathbf{x} + b < 0$. Pode-se observar que a partir de $f(\mathbf{x})$ existem infinitos hiperplanos equivalentes multiplicando $\mathbf{w}$ e b por uma mesma constante. Define-se então o hiperplano canônico em relação ao conjunto T como aquele em que $\mathbf{w}$ e b assumam valores mais próximos de $\mathbf{w} \cdot \mathbf{x} + b = 0$ satisfaçam $|\mathbf{w} \cdot \mathbf{x}_i + b = 1|$ de modo que os classificadores ficam

$$c_j(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0, \forall(\mathbf{x}_i, c_i) \in T. \tag{2.7}$$

Portanto teremos a representação de três hiperplanos, sendo o hiperplano central

$$H = \mathbf{w} \cdot \mathbf{x} + b = 0 \tag{2.8}$$

e os hiperplanos marginais

$$\begin{aligned} H_1 &= \mathbf{w} \cdot \mathbf{x}_i + b = 1 \\ H_2 &= \mathbf{w} \cdot \mathbf{x}_i + b = -1 \end{aligned} . \tag{2.9}$$

Devemos ter a mesma distância $q(\mathbf{w})$ para os vetores que ligam H_1, H_2 a H, chamada de margem e dada por

$$q(\mathbf{x}) = \frac{|\mathbf{w} \cdot \mathbf{x} + b|}{||\mathbf{w}||} = \frac{1}{||\mathbf{w}||}. \tag{2.10}$$

A Figura 2.2 representa o hiperplano H através da reta central contínua, e os dois hiperplanos marginais H_1 e H_2 pelas retas pontilhadas paralelas a H.

Como citado anteriormente, o objetivo é encontrar máx$\{q(\mathbf{x})\}$, semi-retas que ligam as observações até o hiperplano central na Figura 2.2, e podem ser obtidos com a minimização de $||\mathbf{w}||$ ou também $\frac{1}{2}||\mathbf{w}||^2$, logo o problema de encontrar o hiperplano ótimo pode ser formulado como

$$\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2. \tag{2.11}$$

Para garantir a ausência de dados de treinamento entre os vetores de suporte, incluímos na expressão as restrições $c_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0, \forall i = 1, \ldots, n$. Neste caso, trata-se de um problema de programação quadrática convexa (dos Santos, 2015), o qual podemos resolver com base em multiplicadores α_i de Lagrange para encontrar os máximos e mínimos. Então o vetor Normal $\mathbf{w}$ é uma combinação linear de alguns vetores de treino em que $\alpha_i \neq 0$, ou seja, o hiperplano ótimo está em função apenas desses vetores, chamados Vetores de Suporte.

Os conceitos apresentados até então estão limitados para os casos de classificação binária. Para generalizar a técnica de SVM para casos com mais de duas classes, existem alguns métodos propostos, sendo os mais populares o *one-versus-one* e o *one-versus-all*. Suponha ajustar um modelo de classificação com $C > 2$ classes usando SVM. Uma abordagem *one-versus-one* constrói $\binom{C}{2}$ SVMs, e cada um compara um par de classes. Digamos que um SVM compara a k-ésima classe chamada $+1$ com a k'-ésima chamada -1, o que justifica o nome um contra um. O algoritmo classifica uma observação de teste usando cada um dos $\binom{C}{2}$ classificadores e é contado o número de vezes que a observação de teste é atribuída para cada uma das C classes. A classificação final é realizada atribuindo a observação de teste para a classe na qual foi atribuída nestas $\binom{C}{2}$ classificações em pares (James et al., 2013). Maiores detalhes teóricos sobre o SVM podem ser consultados em Krulikovski et al. (2017)

O hiperparâmetro para este método na forma linear é conhecido como custo (Hastie et al., 2009). Basicamente é um mecanismo que impõe uma penalidade ao modelo ao cometer um erro, então quanto maior for o valor deste parâmetro, menor é a probabilidade do algoritmo classifique incorretamente uma observação (Meyer et al., 2019).

2.7 Métodos Baseados em Árvores de Decisão

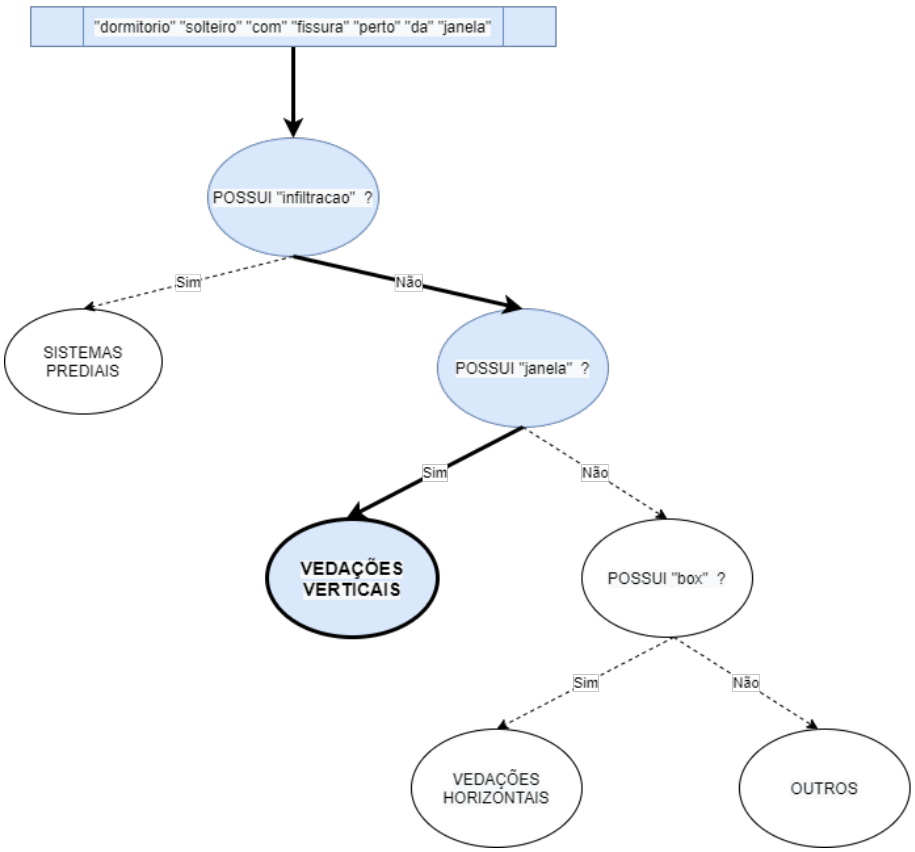
Os métodos baseados em árvores para regressão ou classificação envolvem a segmentação do espaço dos preditores em um número de regiões. Para fazer uma predição dada uma observação geralmente é usada a média ou a moda das observações de

treinamento da região na qual estas pertencem. O conjunto de regras de separação usadas para segmentar o espaço de preditores pode ser resumido e representado em uma árvore, esta abordagem é chamada de métodos de árvore de decisão (James et al., 2013).

Para problemas com respostas qualitativas, são usadas as árvores de classificação, cuja predição é feita de modo que cada observação pertença à classe onde as observações de treino pertenceram de forma mais recorrente (James et al., 2013). Estas árvores são criadas a partir da partição recursiva de um espaço de atributos, a ideia é modelar uma sequência de decisões descrevendo graficamente sua trajetória levando em conta suas combinações de decisões e eventos (dos Santos, 2015). Os elementos que compões uma árvore de decisão são a raiz, nós e folhas. Os nós representam os testes lógicos que dividem o espaço de atributos. A raiz é o primeiro nó, não possui aresta de entrada e representa o começo do problema. Os demais nós possuem uma aresta de entrada cada, sendo aqueles que possuem aresta de saída conhecidos como nós internos. As folhas são os nós terminais, onde o problema termina e a decisão é tomada pelo algoritmo (dos Santos, 2015).

O método de árvores de decisão aplicado a classificação textual pode ser exemplificado inspirado no contexto da construção civil na Figura 2.3 e na Figura 2.4.

Figura 2.3: Árvore de decisão para classificação de texto.



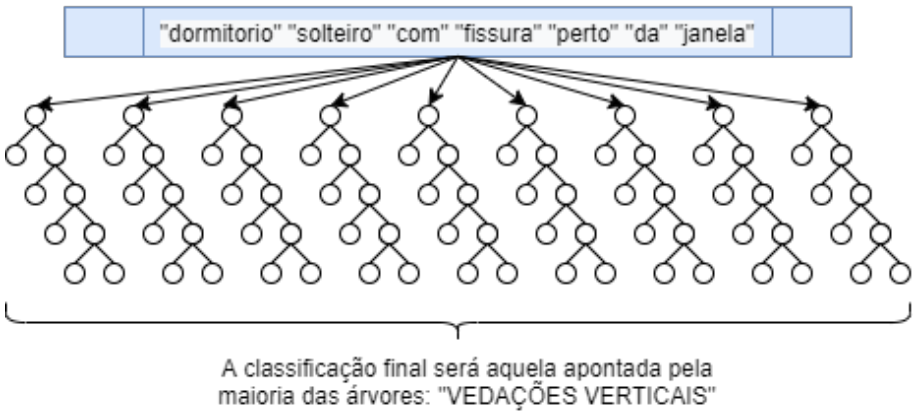
Métodos baseados em árvores são simples e de fácil interpretação. Algoritmos como *Random Forest* e *Boosting* envolvem a criação de múltiplas árvores de decisão que atuam em conjunto para produzir em consenso uma única predição. O uso combinado de um grande número de árvores de decisão tendem a resultar em melhorias

na acurácia do modelo.

2.8 *Random Forest*

O algoritmo *Random Forest* constrói uma sequência de árvores de decisão a partir de um método de reamostragem chamado *bootstrap*, que segundo James et al. (2013) é uma poderosa ferramenta estatística amplamente aplicável em métodos de *statistical learning*. Suponha uma base composta pela totalidade de m palavras, a construção de cada árvore se dá pela seleção aleatória de $s \approx \sqrt{m}$ palavras para composição dos nós e folhas das estruturas (James et al., 2013). O fato de fazer o uso restritivo de s termos ajudará nos casos em que existe um grande número de preditores correlacionados na base de dados. Já o tamanho da floresta é um hiperparâmetro do modelo, sendo que florestas maiores custam maiores tempos de processamento. Após a construção da floresta, os registros a serem classificados passam por todas as árvores e recebem de cada uma delas uma classificação. A categorização final será aquela que for apontada pela maioria das árvores de decisão.

Figura 2.4: Floresta Aleatória.



2.9 *Boosting*

Levando em consideração que no caso do *Random Forest* as árvores de decisão são independentes entre si, o *Boosting* funciona de maneira parecida porém não envolve amostra *bootstrap*. A construção de cada árvore é feita de forma sequencial usando informações das árvores antecessoras, de modo que cada nova árvore é ajustada a partir de uma versão já modificada da base original de dados (James et al., 2013).

Diferentemente de ajustar uma grande árvore com muitos nós, o que resultaria em um ajuste rígido e com potencial problema de *overfitting*, a ideia por trás da técnica é uma abordagem conhecida como de aprendizagem lenta. Esta abordagem consiste em ajustar novas árvores levando em consideração os resíduos do modelo ajustado até o momento da criação da árvore atual. Isso permite que cada uma dessas árvores seja reduzida em termos de nós terminais, e assim gradativamente é melhorada a função $\hat{f}$ ao atacar principalmente as áreas de baixo desempenho preditivo.

Conforme Hastie et al. (2009), um classificador fraco é considerado aquele cuja taxa de erro é ligeiramente melhor do que um palpite aleatório. O objetivo da técnica de *boosting* é aplicar o algoritmo de classificação fraca repetidamente nas versões modificadas dos dados, de tal forma a produzir uma sequência de classificadores fracos. Nota-se que diferentemente do *Random Forest*, a construção de cada árvore depende fortemente das que foram construídas anteriormente, e conforme as iterações avançam, as observações mais difíceis de serem classificadas corretamente recebem pesos de influência cada vez maiores. Portanto cada árvore de decisão na sequência é forçada a se concentrar nas observações de treinamento cujo o modelo não conseguiu classificar corretamente, ou seja os erros.

Segundo James et al. (2013) Os hiperparâmetros para este algoritmo são:

- O número de árvores;
- Encolhimento, chamado no software de *shrinkage*, que controla a taxa de aprendizado. É um número positivo e pequeno geralmente variando entre $[0,001; 0,01]$;
- Profundidade, número de nodos que avançam em cada árvore.

Assim como no *Random Forest* as predições de todos esses classificadores são agrupados para uma predição final, porém, neste caso, a votação conta com a ponderação dos erros aplicada pelo algoritmo.

3 Resultados

As seções deste capítulo apresentarão inicialmente o banco de dados, análise descritiva, classificação dos termos quanto a sua importância para as classes, bem como os resultados dos experimentos executados dos algoritmos de classificação.

3.1 Banco de dados

Os dados utilizados neste trabalho são provenientes do estudo antecessor desenvolvido por Bazzan (2019), cuja a base foi fornecida pela própria empresa em estudo, por conta de sua parceria estabelecida com o Núcleo Orientado para a Inovação da Edificação da Universidade Federal do Rio Grande do Sul (NORIE-UFRGS). O banco de dados fornece reclamações textuais dos clientes de 30 empreendimentos coletadas no departamento de assistência técnica da empresa registrados no período de janeiro de 2017 a março de 2018.

Antes de iniciar as análises das reclamações, é importante indicar a maneira que esses dados serão organizados. Neste sentido o primeiro conceito necessário é o de *token*, que é definido por Silge e Robinson (2017) como sendo uma unidade do texto, como uma palavra, que haja interesse em ser usada para análise do conjunto textual.

Segundo Wickham et al. (2014), usar o formato conhecido como *tidy data* é uma forma mais fácil e mais efetiva de manipulação de dados, e possui uma estrutura definida como:

- Cada variável é uma coluna;
- Cada observação é uma linha;
- Cada tipo de unidade observacional é uma tabela.

Se tratando de dados textuais pode-se definir como formato *tidy* uma tabela contendo um *token*, um parágrafo ou uma frase por linha e os termos nas colunas (Silge e Robinson, 2017).

3.1.1 Pré-processamento dos dados

A preparação dos dados é uma tarefa que ocorre normalmente em várias partes do processo de mineração de texto. O passo imediato após a importação do banco, é o tratamento inicial dos textos de reclamações, que consiste em

- Transformar todos os caracteres em minúsculos;

- Retirar acentuação, números e símbolos;
- Corrigir alguns termos que possam poluir as análises, e identificados com alguma frequência nas observações, como abreviações e palavras concatenadas por erro de digitação.

Todas as transformações aplicadas nesta primeira etapa foram executadas a partir da função *mutate* do pacote *dplyr*, em conjunto com as funções *tolower*, *str_replace*, *incov* e *str_remove_all*.

3.1.2 *Corpus*, *Tokenização* e Representação Matricial

A estruturação dos dados é feita com um objeto do tipo *corpus* que recebe um registro por linha e permite associação de alguma característica ou detalhamento, que para o caso deste trabalho será atribuída a variável categórica *Sistema*. A *tokenização* é o processo de separação dos textos em *tokens* e outros tratamentos, que podem incluir a utilização apenas dos radicais das palavras ou a remoção de palavras consideradas irrelevantes chamadas de *stop words* (Jurasky e Martin, 2000).

Posteriormente este *corpus* poderá ser representado matricialmente por uma *Document Feature Matrix* (DFM), as linhas da matriz são compostas pelos registros, as colunas representam todas as palavras consideradas na análise e os elementos da matriz recebem a frequência que cada palavra aparece no registro (Silge e Robinson, 2017). Para exemplificar estes conceitos, considerando uma base de dados composta por dois registros: *"Parede com defeito."* e *"Defeito na janela."*, o *corpus tokenizado* pelos radicais seria representado pela Tabela 3.1 e a DFM representada pela Tabela 3.2.

Tabela 3.1: Exemplo de *Corpus* tokenizado.

Corpus	Sistema
'pared' 'com' 'defeit'	Estrutura
'defeit' 'na' 'janel'	Esquadrias

Tabela 3.2: Exemplo de *DFM*.

Registro	pared	com	defeit	na	janel
texto 1	1	1	1	0	0
texto 2	0	0	1	1	1

3.1.3 *Biterms* e Variáveis Sintéticas

Segundo Yan et al. (2013) *Biterms* são conjuntos de duas palavras co-ocorrentes em um mesmo contexto, por exemplo em um mesmo curto trecho de texto. Foi considerado os *Biterms* contidos nas reclamações, sendo que selecionados os casos que apareceram no mínimo dez vezes entre as reclamações, neste caso foi apurado um total de 113 duplas de palavras mais frequentes em conjunto. Supondo uma influência elevada destes elementos, foi testada a inclusão de termos concatenados naqueles registros que contenham um ou mais dos *Biterms* identificados no banco.

Tabela 3.3: Exemplo de *Biterms*.

Biterm	Frequência
banheiro suite	168
dormitorio solteiro	106
azulejos soltos	53
box banheiro	14

Tomando como exemplo a Tabela 3.3, que representa uma lista reduzida de *biterms* contidos na base de dados, observar que o registro *"impermeabilizacao box banheiro suite"* possui dois *biterms* da lista, portanto com a inclusão das variáveis sintéticas, este caso seria transformado no banco de teste para *"impermeabilizacao box banheiro suite banheirosuite boxbanheiro"*.

Foi pensada também em uma forma de facilitar a classificação da classe *Estrutura*, uma vez que esta é a menos frequente nas reclamações, mas talvez a de maior interesse de predição por parte da empresa por se tratar de problemas mais graves. A abordagem foi a utilização de algumas palavras que Bazzan (2019) cita como importantes para esta classe. A transformação dos registros se deu então com a inclusão do vetor com as palavras *"junta"*, *"dilatacao"*, *"viga"*, *"pilar"*, *"escada"*.

Outras tentativas de melhoria dos dados, foram inclusões de palavras nas reclamações, assim como nos casos anteriores, porém agora foram incluídos termos mais frequentes, bem como aqueles termos melhores ranqueados pelo índice *Keyness*.

3.1.4 *Oversampling*

Ao lidar com grandes diferenças de tamanhos entre as classes, as técnicas de classificação podem apresentar falhas, que quase sempre resultam em desvios em favor de classes maiores, ou dar pouca relevância para classes menores a ponto de tratá-las como ruído (Kaur e Gosain, 2018). Neste trabalho, existe uma discrepância grande no número de ocorrências entre as classes. A partir da Tabela 3.4, pode-se observar que a classe *Estrutura* é a menos frequente possuindo apenas 30 observações, enquanto que a classe *Sistemas Prediais* aparece 914 vezes, o que sugere a utilização de alguma técnica para dados desbalanceados.

Tabela 3.4: Frequência das classes no banco de dados.

Classe	Ocorrências
COMPLEMENTARES	67
ESQUADRIAS	729
ESTRUTURA	30
SISTEMAS PREDIAIS	914
VEDACOES HORIZONTAIS	450
VEDACOES VERTICAIS	575

Dentre as abordagens para equalizar a base quanto as classes, foi utilizada a técnica chamada *Oversampling*, que é o processo de aumentar o número de observações na classe minoritária, através da replicação aleatória dos mesmos dados ou pela geração de dados sintéticos. A vantagem na utilização desta abordagem é que não há perda de dados para melhorar a razão de desbalanceamento (Kaur e Gosain, 2018). Para balancear o banco foi utilizada a função *upSample* do pacote *caret*.

3.1.5 Partição do banco

A totalidade de registros foi dividida em dois conjuntos conhecidos como bancos de treino e teste (Hastie et al., 2009). Uma das partes separada corresponde a 70% do total da base e será usada para treinamento dos modelos de classificação, já os 30% restantes dos registros serão usados para testar a modelagem ajustada. Esta partição tem como objetivo evitar o problema de *overfitting*, ou seja, não conseguir generalizar o modelo para dados além do banco de treinamento. A seleção dos casos para compor cada parte é feita de forma randomizada, ao fazer uma amostra aleatória de registros da base completa a partir da função *sample_frac* para selecionar registros para treino, e a função *anti_join* para alocar os registros restantes para o grupo de teste.

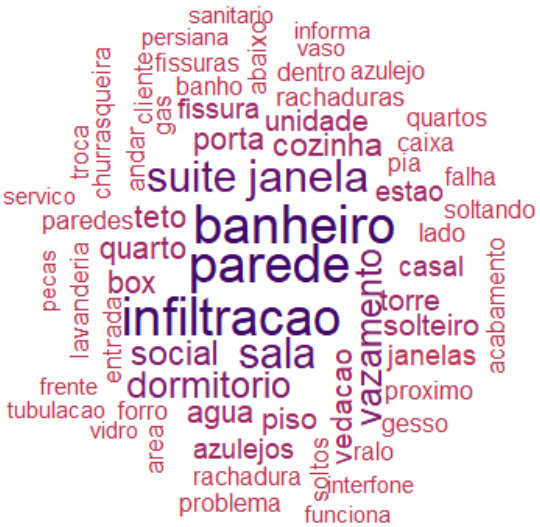
A separação de parte dos dados para validação não foi feita pelo fato de que algumas classes apresentam poucos registros, o que inviabilizaria uma nova segmentação para este fim.

3.2 Análises Descritivas

Para as análises descritivas, a base passou pelo mesmo tratamento dos dados, exceto pela manutenção dos prefixos e sufixos e a retirada de *stop words* para evitar poluir os gráficos com dados não informativos.

A Figura 3.1 foi gerada a partir da lista de palavras que aparecem no mínimo cinquenta vezes no banco de dados, e representa a frequência geral da base. Chamada de nuvem de palavras, os termos maiores e com tonalidade mais escura apresentam maior relevância no *corpus*.

Figura 3.1: Nuvem de Palavras

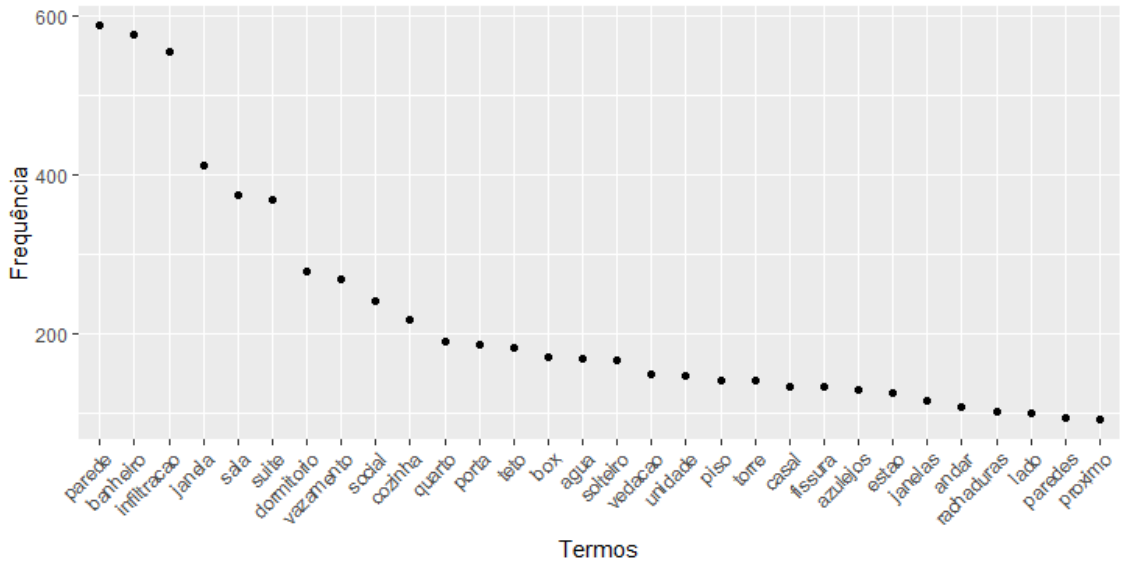


A Tabela 3.5 e a Figura 3.2 apresentam as palavras mais frequentes no banco.

Tabela 3.5: Dez termos mais frequentes.

Rank	Termo	Frequência
1	parede	588
2	banheiro	578
3	infiltracao	556
4	janela	412
5	sala	374
6	suite	368
7	dormitorio	278
8	vazamento	269
9	social	241
10	cozinha	218

Figura 3.2: Vinte e cinco termos mais frequentes



3.3 Análise de importância de *Keyness*

Para exemplificar o índice de *Keyness*, a Figura 3.3 mostra nas barras azuis os dez termos mais importantes para a classe *Complementares*, e faz uma comparação com os termos menos relevantes para esta classe representados pelas barras cinzas, segundo o teste Qui-Quadrado utilizado para esta estatística.

Figura 3.3: Plotagem de *Keyness* para *Complementares*

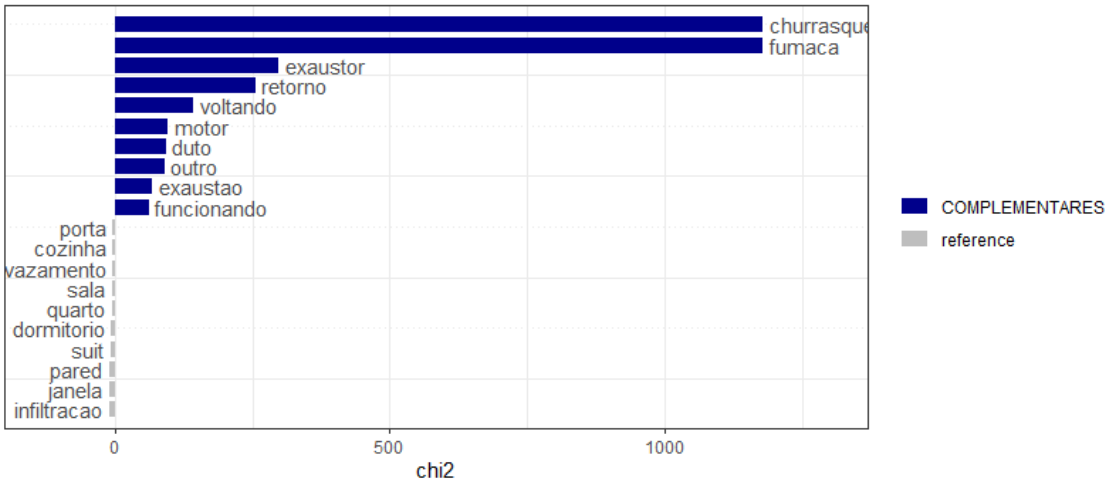


Figura 3.4: Plotagem de *Keyness* para *Esquadrias*

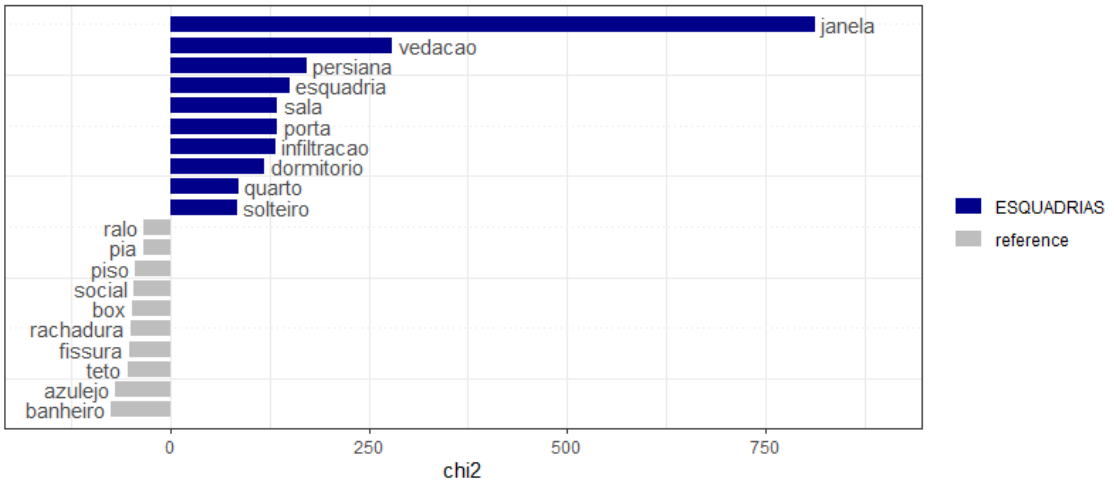


Figura 3.5: Plotagem de *Keyness* para *Estrutura*

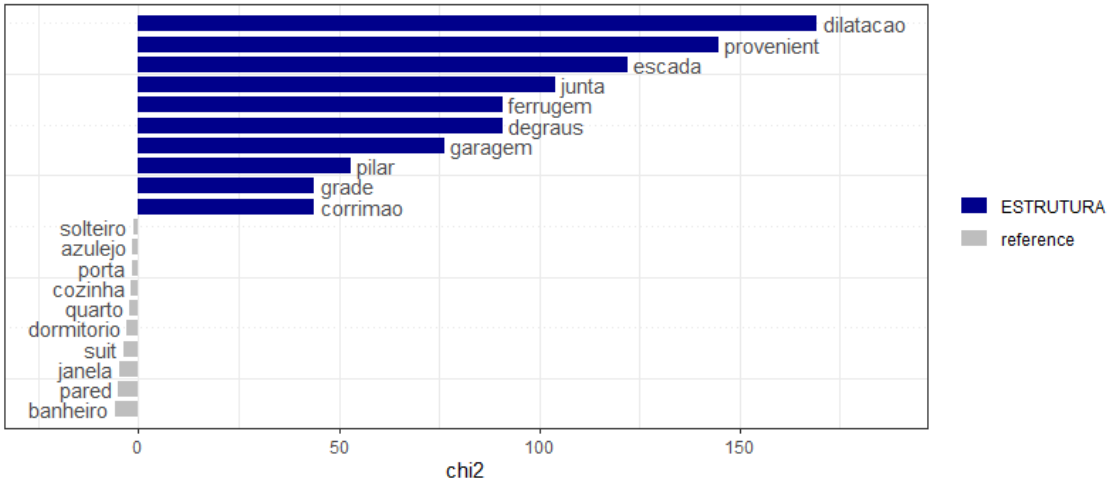


Figura 3.6: Plotagem de *Keyness* para *Sistemas Prediais*

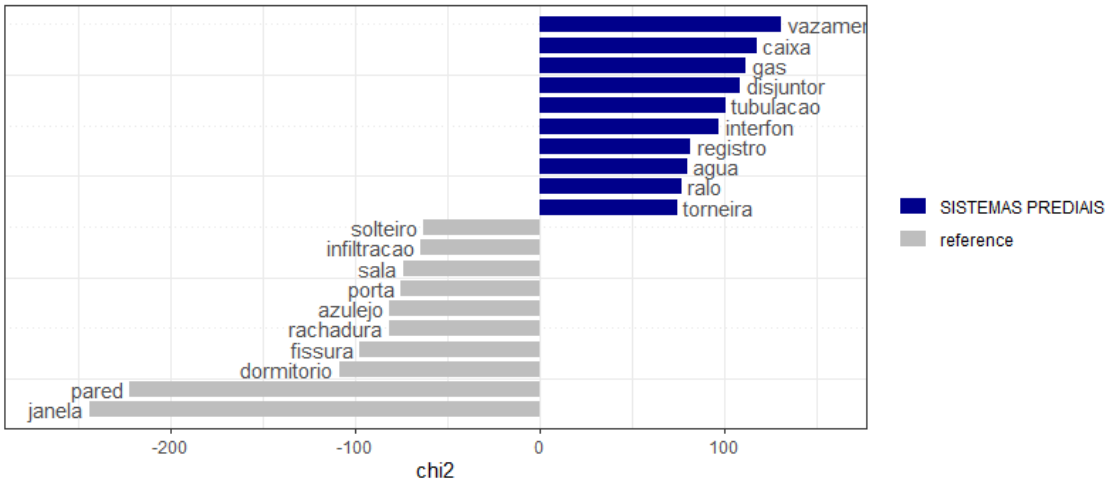


Figura 3.7: Plotagem de *Keyness* para a classe *Vedações Horizontais*

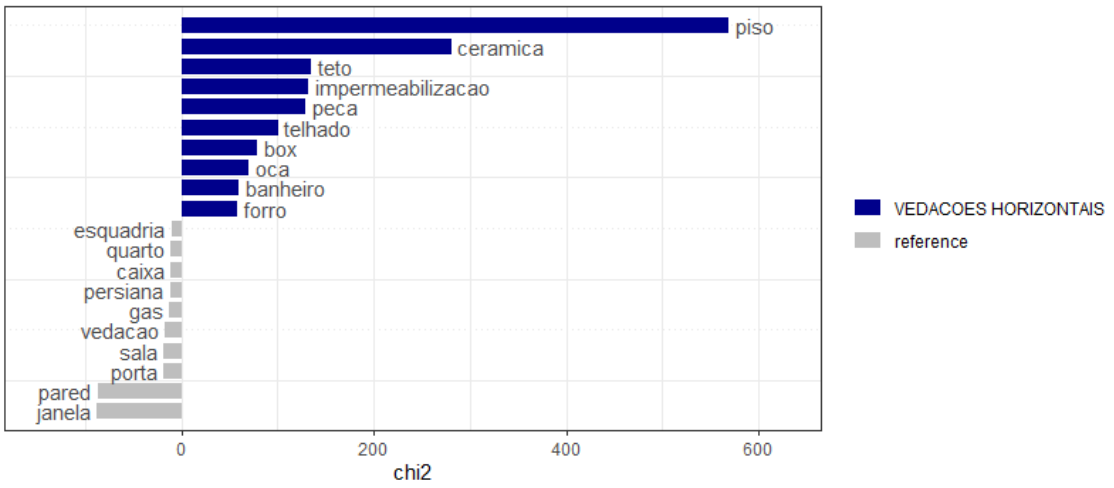
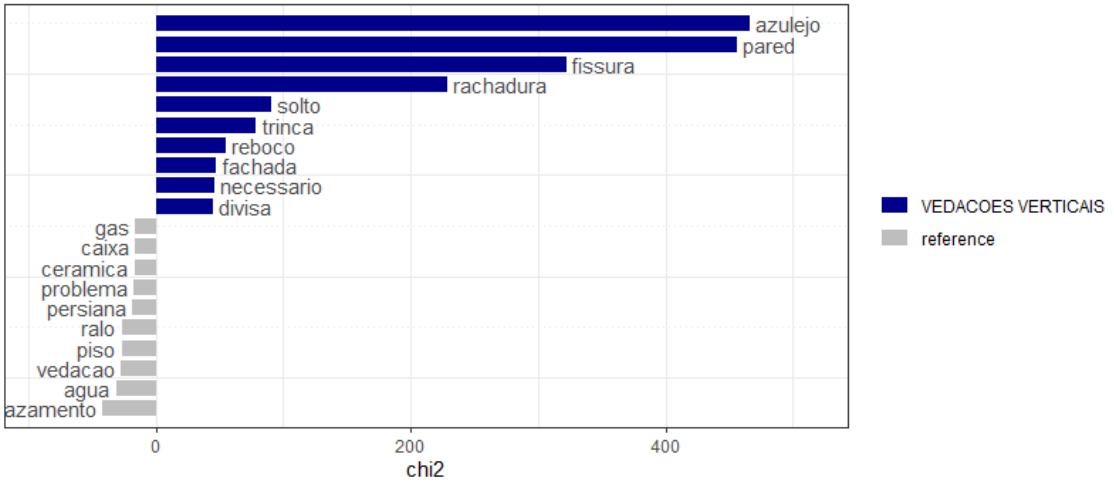


Figura 3.8: Plotagem de *Keyness* para a classe *Vedações Verticais*



A comparação feita na Tabela 3.6, mostra que nem todos os casos mais frequentes

possuem maior relevância para uma determinada classe. Um exemplo é o termo *"parede"* que tem mais do que o dobro de observações do que a segunda palavra mais frequente em *Vedações Verticais*. Porém este mesmo termo é o segundo colocado no quesito importância para esta classe, se ponderada conjuntamente com toda a base. Este fenômeno acontece pois esta palavra aparece muitas vezes também em outras categorias.

Tabela 3.6: Comparação entre Frequência e Importância

Classe	Rank	Frequência	n	Importância	χ^2
Complementares	1	churrasqueira	45	churrasqueira	1178,82
	2	fumaca	26	fumaca	1178,44
	3	exaustor	11	exaustor	297,70
	4	retorno	9	retorno	256,05
	5	funcionando	8	voltando	143,37
Esquadrias	1	janela	426	janela	812,64
	2	infiltracao	266	vedacao	279,62
	3	pared	235	persiana	171,29
	4	sala	200	esquadria	149,99
	5	dormitorio	166	sala	135,20
Estrutura	1	infiltracao	13	dilatacao	169,28
	2	torr	9	provenient	144,57
	3	vazamento	7	escada	121,94
	4	fissura	6	junta	103,88
	5	escada	6	degraus	90,78
Sistemas Prediais	1	banheiro	215	vazamento	131,03
	2	vazamento	174	caixa	117,54
	3	agua	108	gas	111,61
	4	social	94	disjuntor	108,62
	5	infiltracao	91	tubulacao	100,71
Vedações Horizontais	1	banheiro	164	piso	569,04
	2	piso	134	ceramica	280,93
	3	suit	90	teto	135,22
	4	teto	85	impermeabilizacao	132,46
	5	infiltracao	78	peca	129,54
Vedações Verticais	1	pared	382	azulejo	466,00
	2	azulejo	166	pared	455,88
	3	banheiro	158	fissura	322,04
	4	fissura	153	rachadura	228,44
	5	rachadura	124	solto	90,40

3.4 Modelagem

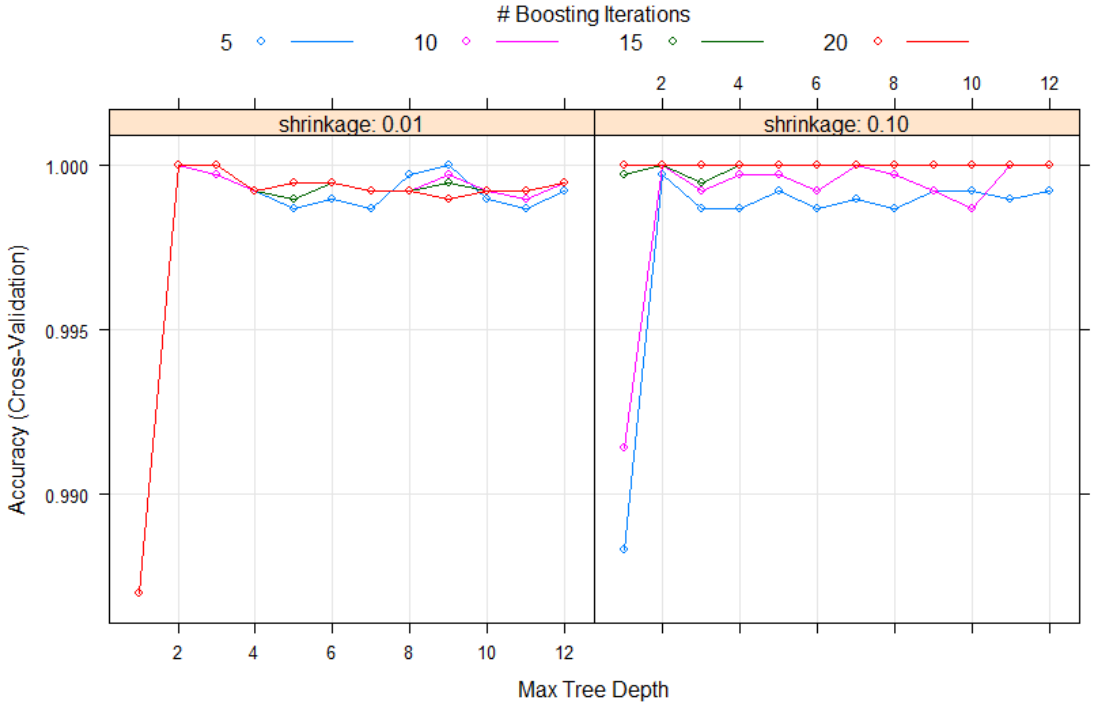
A apresentação a seguir é produto final dos melhores resultados de várias rodadas de modelagem aplicadas a diferentes cenários. As formas de treinamento foram levadas em consideração as variações:

- Partição do banco em treino e teste com respectivas proporções 70%/30%, 80%/20% e 90%/10%;

- Com e sem a exclusão de *stop words*;
- Utilização de abordagens para representação matricial *Bag of Words* e *Word Embeddings* (TF-IDF);
- Utilização ou não de validação cruzada;
- Inclusão de termos sintéticos em observações com a presença de *biterms*;
- Inclusão das palavras mais importantes, segundo avaliação prévia das ocorrências na classe *Estrutura* em todos os registros desta classe;
- Inclusão de um a cinco termos mais frequentes para cada classe;
- Inclusão de um a cinco termos mais bem ranqueados pela estatística *keyness*.

A melhor configuração encontrada para os dados foi a partição de treino e teste com 70%/30%, para que os dados de validação tivessem maior representatividade das classes menos frequentes. Optou-se pela manutenção das *stop words* pela ligeira melhora na precisão geral dos classificadores e a abordagem *Bag of Words*. Foi decidida pela não inclusão de qualquer variável sintética nos registros, pois a intenção de tornar as observações mais informativas acabou por gerar o problema de *overfitting*, como pode-se observar na Figura 3.9 a rápida tendência da precisão atingir valores próximos de 1 para os dados de treinamento, independente de valores de hiperparâmetros.

Figura 3.9: Gráfico de ajuste com *overfitting*

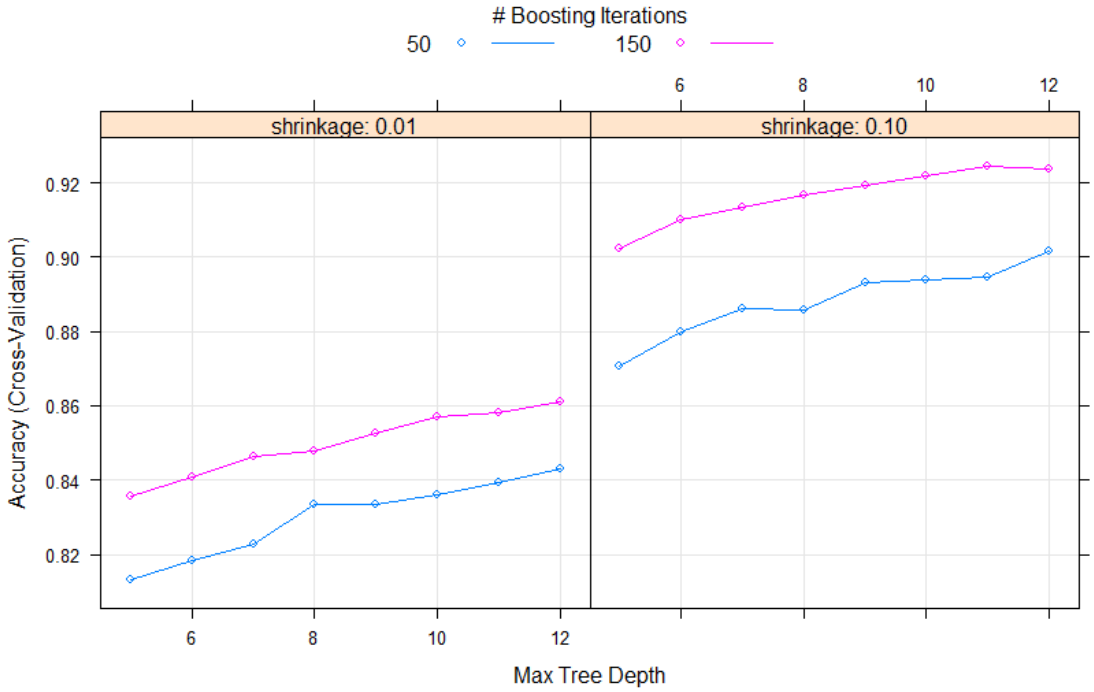


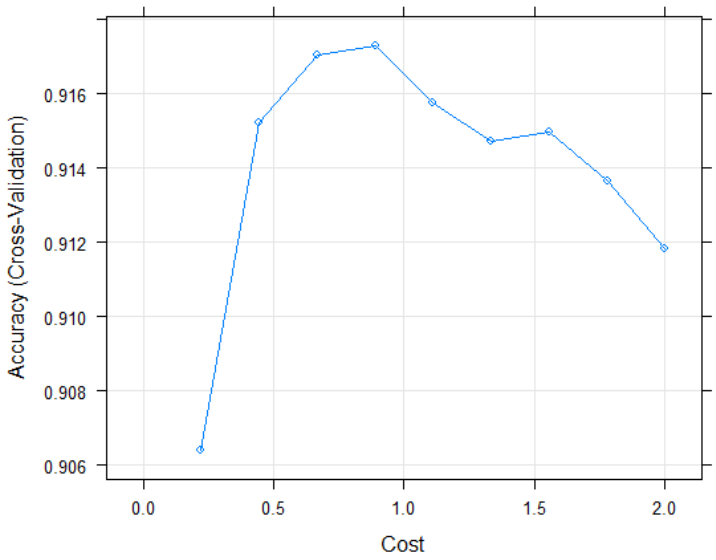
Algumas seleções de diferentes hiperparâmetros serão comparadas em conjunto com os modelos. Para isso, o modelo *Random Forest* foi treinado em dois cenários

com uma grande amplitude no número de árvores. Já para decidir os hiperparâmetros de *SVM* e *Boosting*, primeiramente foi passado um intervalo de argumentos para função de treino para que o software treine as possibilidades informadas de forma iterativa. Após análise da Figura 3.10 e ponderação por tempo de execução, foi selecionado o custo de $0,89$ para *SVM* e os seguintes valores de entrada para o *Boosting*:

- Número de árvores = 150;
- Profundidade de interação = 11;
- Encolhimento = $0,1$

Figura 3.10: Gráficos de *Tuning*
[*Boosting*]





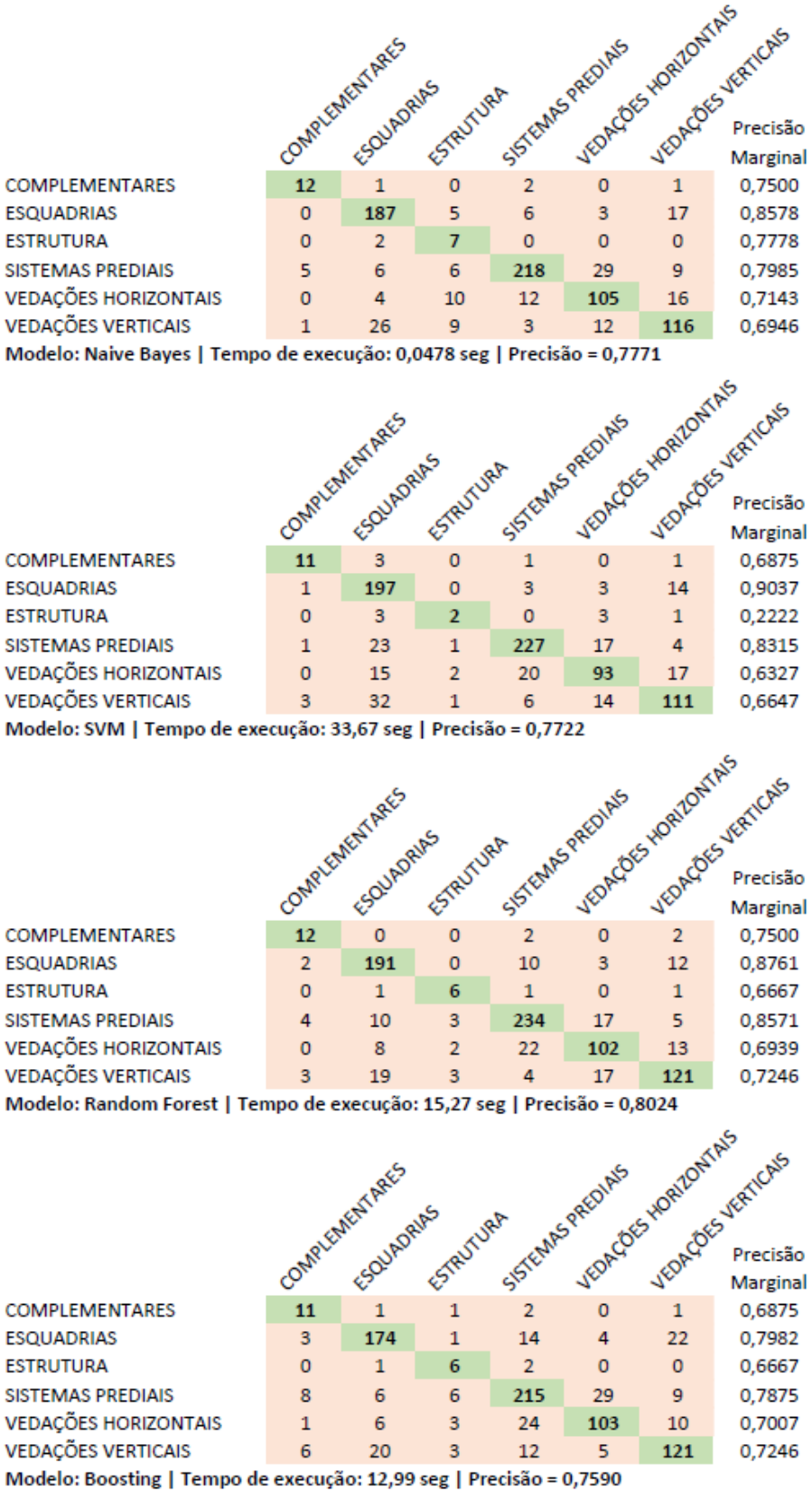
[*SVM*]

Após os ajustes dos modelos, foram criadas matrizes formadas pelos valores reais nas linhas e valores preditos nas colunas. Esta matriz é chamada de matriz de confusão, e é uma forma de avaliar a quantidade de reclamações classificadas corretamente para cada classe. Conjuntamente com dados brutos da matriz confusão, foram calculadas as precisões de cada categoria, bem como a precisão geral do modelo. Precisão é definida como previsões verdadeiras e positivas dividido pelo número total de amostra, portanto o cálculo retornará o percentual de classificações acertadas do modelo.

A Figura 3.11 apresenta as matrizes de confusão para os modelos ajustados inicialmente sem validação cruzada ou seleção de hiperparâmetros. As diagonais

principais das matrizes, células destacadas em verde, representam os valores que os modelos acertaram. As precisões gerais dos modelos variaram pouco, sendo o pior desempenho de com valor de $0,7590$ para *Boosting*, que foi submetido à parametrização padrão do software dos hiperparâmetros. E o melhores resultados para o *Random Forest* com $0,8024$ de precisão geral com a utilização de dez árvores.

Figura 3.11: Matrizes de Confusão: Sem Validação Cruzada



	COMPLEMENTARES	ESQUADRIAS	ESTRUTURA	SISTEMAS PREDIAIS	VEDAÇÕES HORIZONTAIS	VEDAÇÕES VERTICAIS	Precisão Marginal
COMPLEMENTARES	12	1	0	2	0	1	0,7500
ESQUADRIAS	0	187	5	6	3	17	0,8578
ESTRUTURA	0	2	7	0	0	0	0,7778
SISTEMAS PREDIAIS	5	6	6	218	29	9	0,7985
VEDAÇÕES HORIZONTAIS	0	4	10	12	105	16	0,7143
VEDAÇÕES VERTICAIS	1	26	9	3	12	116	0,6946

Modelo: Naive Bayes | Tempo de execução: 0,0478 seg | Precisão = 0,7771

	COMPLEMENTARES	ESQUADRIAS	ESTRUTURA	SISTEMAS PREDIAIS	VEDAÇÕES HORIZONTAIS	VEDAÇÕES VERTICAIS	Precisão Marginal
COMPLEMENTARES	11	3	0	1	0	1	0,6875
ESQUADRIAS	1	197	0	3	3	14	0,9037
ESTRUTURA	0	3	2	0	3	1	0,2222
SISTEMAS PREDIAIS	1	23	1	227	17	4	0,8315
VEDAÇÕES HORIZONTAIS	0	15	2	20	93	17	0,6327
VEDAÇÕES VERTICAIS	3	32	1	6	14	111	0,6647

Modelo: SVM | Tempo de execução: 33,67 seg | Precisão = 0,7722

	COMPLEMENTARES	ESQUADRIAS	ESTRUTURA	SISTEMAS PREDIAIS	VEDAÇÕES HORIZONTAIS	VEDAÇÕES VERTICAIS	Precisão Marginal
COMPLEMENTARES	12	0	0	2	0	2	0,7500
ESQUADRIAS	2	191	0	10	3	12	0,8761
ESTRUTURA	0	1	6	1	0	1	0,6667
SISTEMAS PREDIAIS	4	10	3	234	17	5	0,8571
VEDAÇÕES HORIZONTAIS	0	8	2	22	102	13	0,6939
VEDAÇÕES VERTICAIS	3	19	3	4	17	121	0,7246

Modelo: Random Forest | Tempo de execução: 15,27 seg | Precisão = 0,8024

	COMPLEMENTARES	ESQUADRIAS	ESTRUTURA	SISTEMAS PREDIAIS	VEDAÇÕES HORIZONTAIS	VEDAÇÕES VERTICAIS	Precisão Marginal
COMPLEMENTARES	11	1	1	2	0	1	0,6875
ESQUADRIAS	3	174	1	14	4	22	0,7982
ESTRUTURA	0	1	6	2	0	0	0,6667
SISTEMAS PREDIAIS	8	6	6	215	29	9	0,7875
VEDAÇÕES HORIZONTAIS	1	6	3	24	103	10	0,7007
VEDAÇÕES VERTICAIS	6	20	3	12	5	121	0,7246

Modelo: Boosting | Tempo de execução: 12,99 seg | Precisão = 0,7590

A próxima apresentação refere-se a tentativa de melhoria dos modelos *SVM*, *Random Forest* e *Boosting* através do treinamento com aplicação do *K-fold Cross Validation*. Para os dois últimos modelos com os hiperparâmetros definidos anterior-

mente. Pode-se observar na Figura 3.12 que o *SVM* teve perda de desempenho geral, enquanto que os dois métodos baseados em árvores de decisão foram melhorados.

Figura 3.12: Matrizes de Confusão: Com Validação Cruzada e *tuning*

	COMPLEMENTARES	ESQUADRIAS	ESTRUTURA	SISTEMAS PREDIAIS	VEDAÇÕES HORIZONTAIS	VEDAÇÕES VERTICAIS	Precisão Marginal
COMPLEMENTARES	**12**	2	0	1	0	1	0,7164
ESQUADRIAS	1	**198**	0	3	4	12	0,9045
ESTRUTURA	0	2	**2**	1	3	1	0,2169
SISTEMAS PREDIAIS	1	25	0	**223**	18	6	0,8144
VEDAÇÕES HORIZONTAIS	0	14	2	22	**94**	15	0,6367
VEDAÇÕES VERTICAIS	3	26	1	5	14	**118**	0,7036

Modelo: SVM | Tempo de execução: 5,24 min | Precisão = 0,7795

	COMPLEMENTARES	ESQUADRIAS	ESTRUTURA	SISTEMAS PREDIAIS	VEDAÇÕES HORIZONTAIS	VEDAÇÕES VERTICAIS	Precisão Marginal
COMPLEMENTARES	**12**	1	0	2	0	1	0,7500
ESQUADRIAS	2	**189**	0	9	4	14	0,8670
ESTRUTURA	0	1	**6**	1	0	1	0,6667
SISTEMAS PREDIAIS	3	12	0	**231**	19	8	0,8462
VEDAÇÕES HORIZONTAIS	0	6	1	21	**106**	13	0,7211
VEDAÇÕES VERTICAIS	2	18	1	4	8	**134**	0,8024

Modelo: Random Forest | Tempo de execução: 47,32 min | Precisão = 0,8168

	COMPLEMENTARES	ESQUADRIAS	ESTRUTURA	SISTEMAS PREDIAIS	VEDAÇÕES HORIZONTAIS	VEDAÇÕES VERTICAIS	Precisão Marginal
COMPLEMENTARES	**10**	1	1	3	0	1	0,6250
ESQUADRIAS	2	**193**	0	6	4	13	0,8853
ESTRUTURA	0	2	**5**	1	1	0	0,5556
SISTEMAS PREDIAIS	1	6	1	**238**	20	7	0,8718
VEDAÇÕES HORIZONTAIS	0	6	0	12	**109**	20	0,7415
VEDAÇÕES VERTICAIS	2	14	2	5	11	**133**	0,7964

Modelo: Boosting | Tempo de execução: 4,59 hrs | Precisão = 0,8289

Pelas matrizes de confusão da Figura 3.13 percebe-se que o modelo *SVM* não obteve ganho de performance ao utilizarmos o hiperparâmetro de custo apontado com o uso da análise da validação cruzada. No entanto, para os demais modelos, a melhora foi considerável, com destaque para o *Boosting* que atingiu a proporção de acertos de $83,61\%$ para o banco de teste. Ao analisarmos as precisões marginais identificou-se uma baixa proporção de acertos para aquelas classes que possuem menos observações, principalmente a classe *Estrutura*, que possui apenas nove registros para teste. Observa-se também um confundimento geral dos modelos entre as categorias *Esquadrias* e *Vedações Verticais*.

Figura 3.13: Matrizes de Confusão: com *tuning*

	COMPLEMENTARES	ESQUADRIAS	ESTRUTURA	SISTEMAS PREDIAIS	VEDAÇÕES HORIZONTAIS	VEDAÇÕES VERTICAIS	Precisão Marginal
COMPLEMENTARES	12	2	0	1	0	1	0,7500
ESQUADRIAS	1	189	0	5	4	19	0,8670
ESTRUTURA	0	2	2	0	4	1	0,2222
SISTEMAS PREDIAIS	1	29	1	219	19	4	0,8022
VEDAÇÕES HORIZONTAIS	0	14	2	26	89	16	0,6054
VEDAÇÕES VERTICAIS	3	28	1	12	11	112	0,6707

Modelo: SVM | Tempo de execução: 1,96 min | Precisão = 0,7506

	COMPLEMENTARES	ESQUADRIAS	ESTRUTURA	SISTEMAS PREDIAIS	VEDAÇÕES HORIZONTAIS	VEDAÇÕES VERTICAIS	Precisão Marginal
COMPLEMENTARES	12	1	0	2	0	1	0,7500
ESQUADRIAS	2	189	0	9	4	14	0,8670
ESTRUTURA	0	1	6	1	0	1	0,6667
SISTEMAS PREDIAIS	3	12	0	231	19	8	0,8462
VEDAÇÕES HORIZONTAIS	0	6	1	21	106	13	0,7211
VEDAÇÕES VERTICAIS	2	18	1	4	8	134	0,8024

Modelo: Random Forest | Tempo de execução: 11,13 min | Precisão = 0,8168

	COMPLEMENTARES	ESQUADRIAS	ESTRUTURA	SISTEMAS PREDIAIS	VEDAÇÕES HORIZONTAIS	VEDAÇÕES VERTICAIS	Precisão Marginal
COMPLEMENTARES	12	1	0	2	0	1	0,7500
ESQUADRIAS	2	195	0	5	4	12	0,8945
ESTRUTURA	0	2	4	2	1	0	0,4444
SISTEMAS PREDIAIS	0	6	1	240	17	9	0,8791
VEDAÇÕES HORIZONTAIS	0	5	1	16	106	19	0,7211
VEDAÇÕES VERTICAIS	2	15	1	2	10	137	0,8204

Modelo: Boosting | Tempo de execução: 5,68 min | Precisão = 0,8361

Tabela 3.7: Resumo dos modelos.

Modelo	Tempo	Precisão
Naive Bayes	0,04 seg	0,7771
SVM	33,67 seg	0,7722
SVM *tuning* com Validação Cruzada	5,24 min	0,7795
SVM *tuning*	1,96 min	0,7506
RF 10 árvores	15,27 seg	0,8024
RF 200 árvores com Validação Cruzada	47,32 min	0,8168
RF 200 árvores	11,13 min	0,8192
Boosting	12,99 seg	0,7590
Boosting *tuning* com Validação Cruzada	4,59 hrs	0,8289
Boosting *tuning*	5,68 min	0,8361

Observa-se na Tabela 3.7 que os tempos de execução dos modelos baseados em árvores de decisão são mais elevados, principalmente o *Boosting* com validação cruzada. O alto custo computacional neste caso é é explicado pela maior quantidade de hiperparâmetros deste método, gerando um número maior de possíveis combinações de cenários que o software considera para os treinamentos.

4 Conclusão

Através das análises descritivas já é possível extrair informações relevantes sobre os dados, que se forem avaliadas com conhecimento sobre construção civil mais qualificado e voltado para o negócio, podem estimular novas segmentações ou formas de visualizações que produzam maior valor às análises.

A proposta de enriquecer os registros do banco de treinamento com a inclusão de palavras sintéticas tornaram os modelos ineficientes para generalização pelo problema de *overfitting*, principalmente para os dois métodos baseados em árvores de decisão.

O classificador *Naive Bayes* por se tratar de uma abordagem mais simplista, não possui estratégias de *tuning* do modelo, ficando então limitado aos resultados sem o uso da validação cruzada para melhorias de hiperparâmetros. Em contrapartida este método além de rápido processamento apresentou números satisfatórios de precisão, sendo a segunda melhor técnica se comparada com as demais sem refinamentos dos parâmetros. Destacando também pela melhor capacidade de classificar reclamações menos frequentes, como de *Estrutura*, cometendo apenas dois erros no banco de teste, acertando em $0,7778$ dos casos.

Dentre os modelos ajustados com base na validação cruzada, o SVM teve uma performance parecida com o *Naive Bayes*, oscilando a precisão geral entre $0,75$ e $0,78$ mesmo com as tentativas de *tuning*. Além disso, apresentou resultados ruins para a classe *Estrutura*, acertando apenas em torno de $0,22$ dos casos pertencentes a esta categoria.

Quanto ao *Random Forest* e o *Boosting*, ambos tiveram bons resultados e melhoraram seus indicadores com os refinamentos dos parâmetros. Maiores tempos de execução foram observados em comparação com os demais modelos, principalmente quando submetidos a processos iterativos, porém mais suscetíveis ao processo de *tuning*. O destaque fica para o *Boosting* que chegou ao maior percentual geral de acerto de $0,8361$ do banco de treino.

Referências Bibliográficas

Arif-Uz-Zaman, K., Cholette, M. E., Ma, L., e Karim, A. (2017). Extracting failure time data from industrial maintenance records using text mining. *Advanced Engineering Informatics*, 33:388–396.

Basu, A., Walters, C., e Shepherd, M. (2003). Support vector machines for text categorization. In *36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the*, pages 7–pp. IEEE.

Bazzan, J. (2019). Método para coletar e analisar dados de assistência técnica da construção civil.

Chi, N.-W., Lin, K.-Y., e Hsieh, S.-H. (2014). Using ontology-based text classification to assist job hazard analysis. *Advanced Engineering Informatics*, 28(4):381–394.

Chi, S., Suk, S.-J., Kang, Y., e Mulva, S. P. (2012). Development of a data mining-based analysis framework for multi-attribute construction project information. *Advanced Engineering Informatics*, 26(3):574–581.

Cupertino, D. e Brandstetter, M. C. G. d. O. (2015). Proposição de ferramenta de gestão pós-obra a partir dos registros de solicitação de assistência técnica. *Ambiente Construído*, 15(4):243–265.

Das, S. e Chew, M. Y. (2011). Generic method of grading building defects using fmeca to improve maintainability decisions. *Journal of Performance of Constructed Facilities*, 25(6):522–533.

de Castro, M. G. (2017). Utilização de text mining para apoio à classificação de registos de alergias e reações adversas.

de Pádua Moreira, R. e Nascimento, C. L. (2012). Prognostics of aircraft bleed valves using a svm classification algorithm. In *2012 IEEE Aerospace Conference*, pages 1–8. IEEE.

dos Santos, L. S. F. C. (2015). Estudo online da dinâmica espaço-temporal de crimes através de dados da rede social twitter.

Gabrielatos, C. (2018). Keyness analysis: Nature, metrics and techniques. inc. taylor & a. marchi (eds.), corpus approaches to discourse: A critical review (pp. 225–258). *New York, NY: Routledge*, 10:9781315179346–11.

Gunay, H. B., Shen, W., e Yang, C. (2019). Text-mining building maintenance work orders for component fault frequency. *Building Research & Information*, 47(5):518–533.

Hastie, T., Tibshirani, R., e Friedman, J. (2009). *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media.

James, G., Witten, D., Hastie, T., e Tibshirani, R. (2013). *An introduction to statistical learning*, volume 112. Springer.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer.

Jurasky, D. e Martin, J. H. (2000). Speech and language processing: An introduction to natural language processing. *Computational Linguistics and Speech Recognition. Prentice Hall, New Jersey*.

Kaur, P. e Gosain, A. (2018). Comparing the behavior of oversampling and undersampling approach of class imbalance learning by combining class imbalance problem with noise. In *ICT Based Innovations*, pages 23–30. Springer.

Krulikovski, E. H. M., Sachine, M., e Ribeiro, A. A. (2017). Análise teórica de máquinas de vetores suporte. In *II Simpósio de Métodos Numéricos em Engenharia*.

Lantz, B. (2013). *Machine learning with R*. Packt Publishing Ltd.

Magalhães, M. N. e de Lima, A. C. P. (2004). *Noções de probabilidade e estatística*, volume 6. Editora da Universidade de São Paulo.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., e Leisch, F. (2019). *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*. R package version 1.7-3.

R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Santos, K. B. C. d. et al. (2019). Categorização de textos por aprendizagem de máquina.

Silge, J. e Robinson, D. (2017). *Text mining with R: A tidy approach*. "O'Reilly Media, Inc.".

Wang, D., Zhang, H., Liu, R., Lv, W., e Wang, D. (2014). t-test feature selection approach based on term frequency for text categorization. *Pattern Recognition Letters*, 45:1–10.

Wang, Y., Zhou, Z., Jin, S., Liu, D., e Lu, M. (2017). Comparisons and selections of features and classifiers for short text classification. In *IOP Conference Series: Materials Science and Engineering*, volume 261, page 012018. IOP Publishing.

Wickham, H. et al. (2014). Tidy data. *Journal of statistical software*, 59(10):1–23.

Yan, X., Guo, J., Lan, Y., e Cheng, X. (2013). A biterm topic model for short texts. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1445–1456.

Anexo: Algoritmo

```
source("pacotes.R")

# IMPORTAÇÃO E TRATAMENTOS ----
data <- read_excel("banco de dados.xlsx")


# _Pré-tratamentos (criação ID e clean) ----
data <- data %>%
  mutate(clean = tolower(data$‘DESCRIÇÃO RESUMIDA‘)) %>%
  mutate(clean = iconv(data$‘DESCRIÇÃO RESUMIDA‘,
                       from = "UTF-8",
                       to="ASCII//TRANSLIT")) %>%
  mutate(clean = str_replace_all(clean,
                                 "dorm.solteiro", "dormitorio solteiro")) %>%
  mutate(clean = str_replace_all(clean,
                                 "dorm. solteiro", "dormitorio solteiro")) %>%
  mutate(clean = str_replace_all(clean,
                                 "dorm.casal", "dormitorio casal")) %>%
  mutate(clean = str_replace_all(clean,
                                 "dorm. casal", "dormitorio casal")) %>%
  mutate(clean = str_replace_all(clean,
                                 "dorm.cas", "dormitorio casal")) %>%
  mutate(clean = str_replace_all(clean, " da ", " ")) %>%
  mutate(clean = str_replace_all(clean, " das ", " ")) %>%
  mutate(clean = str_replace_all(clean, " de ", " ")) %>%
  mutate(clean = str_replace_all(clean, " do ", " ")) %>%
  mutate(clean = str_replace_all(clean, " dos ", " ")) %>%
  mutate(clean = str_replace_all(clean, " a ", " ")) %>%
  mutate(clean = str_replace_all(clean, " o ", " ")) %>%
  mutate(clean = str_remove_all(clean, "[(.)-+]?")) %>%
  mutate(clean = tolower(clean)) %>%
  mutate(SISTEMA = ifelse(SISTEMA!=’ESQUADRIAS’, SISTEMA,
                          ifelse(ELEMENTO==
                          ’GUARDA CORPO / CORRIMÃO / ESCADAS METÁLICAS’,
                          "ESTRUTURA", "ESQUADRIAS"))) %>%
  mutate(ID = row_number())
```

```
# _Partição do banco - Treino/Teste   ----
set.seed(2006)
train <- data %>%
  sample_frac(0.7)
test <- data %>%
  anti_join(train, by = "ID")


# _Oversampling (somente em treino)
train$SISTEMA <- as.factor(train$SISTEMA)
train <- train %>%
  upSample(train$SISTEMA)


# TOKEN ----
# _Corpus treino ----
tok_train <- train$clean %>%
  corpus()
docvars(tok_train, "Sistema")  <- train$SISTEMA
docvars(tok_train, "ID") <- train$ID


#_Token treino ----
tok_train <- tok_train %>%
  tokens(remove_punct = TRUE,
         remove_symbols = TRUE,
         remove_numbers = TRUE,
         remove_separators = TRUE,
         split_hyphens = TRUE) %>%
  tokens_tolower() %>%
  tokens_wordstem(language = "portuguese")


#_DFM
DFM_train <- tok_train %>%
  dfm(tolower = T)
DFM_train_m <- as.matrix(DFM_train)


# _Corpus TESTE ----
tok_test<- test$clean %>%
  corpus()
docvars(tok_test, "Sistema")  <- test$SISTEMA
docvars(tok_test, "ID") <- test$ID


#_Token teste ----
```

```
tok_test <- tok_test %>%
  tokens(remove_punct = TRUE,
         remove_symbols = TRUE,
         remove_numbers = TRUE,
         remove_separators = TRUE,
         split_hyphens = TRUE) %>%
  tokens_wordstem(language = "portuguese") %>%
  tokens_tolower()


#_DFM
DFM_test <- tok_test %>%
  dfm(tolower = T)
DFM_test_m <- as.matrix(DFM_test)


# TUNNING E CONTROLE DE TREINO ----
gbGrid <-  expand.grid(interaction.depth = 11,
                       n.trees = 150,
                       shrinkage = c(0.1),
                       n.minobsinnode = 12)
control <- trainControl(method= "cv",
                        number = 5,
                        search = "grid",
                        allowParallel = T)


######## MODELAGEM #########
# NAIVE BAYES ----
#_______________________________________________
start_time <- Sys.time()
fit <- textmodel_nb(DFM_train,
                    DFM_train$Sistema,
                    distribution = "multinomial")
total_time <- Sys.time() - start_time

predicted_class <- predict(fit, DFM_test, force = TRUE)
actual_class <- test$SISTEMA # valores reais
tab_class <- table(actual_class, predicted_class)

prop <- vector()
acertos <- 0
for (i in 1:6) {
  prop[i] <- tab_class[i,i]/sum(tab_class[i,])
  acertos <- acertos + tab_class[i,i]
}

tab_class <- cbind(tab_class, prop)
```

```
#_____________________________________________


# RANDOM FOREST ----
#_____________________________________________
start_time <- Sys.time()
set.seed(150)
fit <- train(x = DFM_train_m,
             y = DFM_train$Sistema,
             method = "rf",
             trControl = trainControl("none"),
             ntree = 200) # feito com: c(10, 200)
total_time <-Sys.time() - start_time

predicted_class <- predict(fit, newdata = as.data.frame(DFM_test_m))
actual_class <- DFM_test$Sistema
tab_class <- table(actual_class, predicted_class)

prop <- vector()
acertos <- 0
for (i in 1:6) {
  prop[i] <- tab_class[i,i]/sum(tab_class[i,])
  acertos <- acertos + tab_class[i,i]
}

tab_class <- cbind(tab_class, prop)
#_____________________________________________


# SUPPORT VECTOR MACHINE ----
#_____________________________________________
start_time <- Sys.time()
set.seed(150)
fit <- train(x = DFM_train_m,
             y = DFM_train$Sistema,
             method = "svmLinear2",
             trControl = trainControl("none"),
             tuneGrid = data.frame(cost = 0.8888889))
total_time <-Sys.time() - start_time

predicted_class <- predict(fit, newdata = as.data.frame(DFM_test_m))
actual_class <- DFM_test$Sistema
tab_class <- table(actual_class, predicted_class)

prop <- vector()
acertos <- 0
for (i in 1:6) {
  prop[i] <- tab_class[i,i]/sum(tab_class[i,])
```

```
  acertos <- acertos + tab_class[i,i]
}

tab_class <- cbind(tab_class, prop)
#_____________________________________________


# BOOSTING ----
#_____________________________________________
start_time <- Sys.time()
set.seed(150)
fit <- train(x = DFM_train_m,
             y = DFM_train$Sistema,
             method = "gbm",
             trControl = trainControl("none"),
             tuneGrid = gbGrid)
total_time <-Sys.time() - start_time

predicted_class <- predict(fit, newdata = as.data.frame(DFM_test_m))
actual_class <- DFM_test$Sistema
tab_class <- table(actual_class, predicted_class)

prop <- vector()
acertos <- 0
for (i in 1:6) {
  prop[i] <- tab_class[i,i]/sum(tab_class[i,])
  acertos <- acertos + tab_class[i,i]
}

tab_class <- cbind(tab_class, prop)
#_____________________________________________
```