Universidade Federal de Pernambuco
Centro de Informática

Graduação em Ciência da Computação

Classificação e Predição de Preços de Imóveis a Partir de Dados Estruturados e Não Estruturados

Matheus Barbosa Domingues Feliciano

Proposta de Trabalho de Graduação

Orientador: Prof. Luciano de Andrade Barbosa

Recife
12 de dezembro de 2018

Universidade Federal de Pernambuco
Centro de Informática

Matheus Barbosa Domingues Feliciano

Classificação e Predição de Preços de Imóveis a Partir de Dados Estruturados e Não Estruturados

Trabalho apresentado ao Programa de Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: *Prof. Luciano de Andrade Barbosa*

Recife
12 de dezembro de 2018

Agradecimentos

Agradeço a minha família, que me deu apoio, carinho e força para eu conseguir chegar onde cheguei hoje. Em especial a minha mãe, que sempre me aconselhou, me incentivou a nunca desistir e me deu oportunidade de ter experiências incríveis em minha vida. Ao meu pai que me introduziu no mundo da computação, sempre sendo um intusiasta de novas tecnologias e me dando oportunidade de entrar em contato com elas.

Também agradeço ao meu orientador Luciano Barbosa, por ter me aconselhado e ajudado durante todo o processo da realização desse trabalho e até mesmo antes disso. O aprendizado adquirido através dele me fez enxergar uma nova área de interesse e me dar novas possibilidades para o meu futuro.

Por fim, agradeço a todos os meus amigos e minha namorada pelo apoio emocional e o incentivo a dar o meu melhor. Obrigado a todos.

Expose yourself to your deepest fear; after that, fear has no power, and the fear of freedom shrinks and vanishes. You are free.

—JIM MORISSON (The Doors)

Resumo

O crescimento do uso da internet como o principal meio de consulta e geração de conteúdo, nos últimos anos, se refletiu numa quantidade imensa de dados na web. Esses dados podem ser utilizados de diversas formas, como análises estatísticas, modelagem de dados e previsões. O principal objetivo deste trabalho é estimar o valor de venda de imóveis, a partir de dados estruturados e não-estruturados de anúncios na Web. Para isso, vai se verificar se a utilização de dados não-estruturados para a predição pode melhorar a qualidade da estimativa de valores desses imóveis.

Palavras-chave: ciência de dados, regressão, pré-processamento de dados, aprendizagem de máquina, setor imobiliário, dados estruturados, processamento de texto, dados não-estruturados

Abstract

The rising of internet usage as the main source and generation of data, in the last few years, has made a huge impact in the amount of data in the web. This data can be used in many ways, as statistical analyses, data modeling and prediction. The main objective of the project is to predict, using real estate structured and unstructured data, the price of a given real estate. The real estate business in Brazil is taking a rising course. Due to the crisis in other types of business, real estate is being treated as one of the safest type of investment in the current status. As result of that, comparing what is the impact of combined usage of unstructured (text data) and structured data to predict the value of real estate in precision and other metrics can be of much value.

Keywords: data science, regression, pre-processing data, machine learning, real estate, structured data, text processing, unstructured data

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Introdução

A avaliação imobiliária, que é o processo de estimar o preço de imóveis, é crucial para ambos compradores e vendedores como uma base para a negociação e transação. O mercado imobiliário tem um papel vital em todos os aspectos da nossa sociedade contemporânea [QY17]. Com o crescimento da geração de dados online sobre imóveis, o caminho mais intuitivo tomado para a análise desses foi o de estimar esse preço através dos atributos desses imóveis, ou seja, os dados já estruturados, como quantidade de quartos, localização, área interna, etc.

Porém, outra forma de talvez aumentar a acurácia e precisão de um algoritmo que faça essa estimativa seria utilizar em conjunto com os dados estruturados, dados não-estruturados, neste caso, as descrições dos imóveis, comentários sobre ele e avaliações de usuários. Todos esses são exemplos de dados não-estruturados em forma de texto. Que seriam tratados de forma a inferir características adicionais aos imóveis que poderiam influenciar no seu preço, e dessa forma aumentando a precisão do algoritmo.

A análise de dados não-estruturados, mais especificamente de textos, para a predição de preços de produtos é uma tendência que vem aumentando nos últimos anos. Uma competição promovida pela Kaggle e a Mercari [Mer18], neste ano, teve como foco a construção de um algoritmo que sugeria automaticamente o preço correto de produtos [Mer18], sendo fornecidos dados não-estruturados em forma de textos de descrições escritos por usuários com informações como marca, categoria do produto e condições.

Utilizando essas duas formas de avaliação imobiliária através de algoritmos de análise de dados estruturados e não-estruturados, poderemos comparar a utilização das duas formas de avaliação e como um interfere na precisão e acurácia do outro quando utilizadas em conjunto. Desse modo pretende-se criar uma ferramenta que consiga utilizar os dois métodos separadamente, assim como também em conjunto e compará-los.

CAPÍTULO 2

Fundamentos

Para a construção de um modelo de regressão de qualidade vários processos de tratamento, ajuste e pre-processamento dos dados devem ser feitos a priori. Esses processos são essenciais para garantir o melhor desempenho e qualidade de um modelo de regressão. Neste capítulo 2, iremos abordar alguns desses processos que involvem dados estruturados e não-estruturados. Aprofundando mais no pré-processamento textual.

2.1 Pré-processamento de dados

Antes de utilizar qualquer tipo de dado para construção de um modelo estatístico, aplicação de algoritmo de aprendizagem supervisionada ou mesmo uma análise dos dados, deve haver um tratamento desses dados para que a corretude, qualidade e desempenho dessas tarefas.

A extração dos dados pode vir de várias fontes: bancos de dados, tabelas excel, html, etc. E muitas vezes quando extraídos alguns problemas devem ser resolvidos. Dentre estes estão a decteção e limpeza de dados errônios ou mesmo irrevalentes, como tanto *outliers* - observações ou medidas suspeitas por ser muito menor ou maior do que a maioria das observações coletadas. [DC10]

Nessa seção 2.1, iremos introduzir o conceito de dados estruturados e alguns dos problemas que podem ser resolvidos no pré-processamento para a construção de modelos melhores. Iremos também introduzir o conceito de dados não-estruturados e abordar a extração de textos a partir de páginas da web e o seu pré-processamento.

2.1.1 Dados estruturados

Dados estruturados são caracterizados por serem dados que possuem uma organização, dados que podem ser facilmente acessados diretamente através de tabelas ou bancos de dados. Nesta seção iremos abordar alguns processos de pré-processamento de dados estruturados.

2.1.1.1 Limpeza de dados

Limpeza de dados é um processo de pré-processamento de dados definido por três estágios envolvidos em um ciclo repetitivo: triagem, diagnóstico e edição (ou remoção) de anormalidades. [GJ13]

A fase de triagem tem como objetivo verificar *outliers*, inconsistências, padrões anormais e a falta ou excessão de informação. Uma estratégia para detecção de anomilias é inserir os dados em um tabela onde podem ser visualizados e verificados de diversas maneiras. Inici-

almente, uma análise de consulta dessas tabelas pode ser realizada para verificar se há algum dado discrepante, irrelevante ou faltante, pode ser feita. Como por exemplo, um conjunto de dados que descreve imóveis ter um valor de venda do imóvel menor que zero, ou até mesmo muito baixo em comparação aos outros. Após a análise através de consultas, uma análise estatística através gráficos como *boxplot*, histogramas e gráficos de dispersão pode ser feita para que a visualização dessas anormalidades seja mais evidente para o cientista de dados.

A fase de diagnóstico tem como objetivo clarificar a natureza dos problemas, padrões e estatísticas identificados na fase de triagem. Nessa fase, os diagnósticos podem verificar facilmente um dado impossível (como o valor de venda negativo), suspeito(como valor de venda muito baixo) ou faltante (onde simplesmente não há valor no conjunto de dados para aquela instância).

A fase de tratamento tem como objetivo resolver os problemas analisados nas fases anteriores, tendo como opção corrigir, deletar ou mesmo ignorar os dados anormais. Para dados impossíveis ou faltantes, quando possível encontrar os dados corretos, uma correção deve ser feita, porém se não houver como encontrá-los, é preferível uma deleção. O julgamento para resolução desses problemas varia bastante caso a caso.

2.1.2 Dados não estruturados

Dados não estruturados são conjunto de dados que não possuem uma organização definida. Dados em que as informações não podem ser extraídas diretamente e por isso exigem uma aproximação diferente quanto ao pré-processamento. Nesta seção iremos analisar algumas técnicas de extração e pré-processamento de dados não estruturados, mais especificamente dados textuais.

2.1.2.1 Extração de texto a partir de HTML

A extração de dados não-estruturados (ou semi-estruturados) na maioria das vezes tem como foco as páginas em si, no caso dos textos, o conteúdo da página como um todo. Porém, há também técnicas de mapeamento ou estruturamento desses dados.

A extração pode tanto ser na força bruta usando expressões regulares para extrair informações [ABT16] de uma tag específica ou mesmo de várias tags, quanto abordagens mais sofisticadas.

Uma dessas abordagens é o DOM (Document Object Model) que é um padrão para criação e manipulação de representações de HTML (e XML). Analisando e transcrevendo uma página HTML para uma DOM tree (ou árvore DOM) não só conseguimos extrair grandes unidades lógicas, como também links específicos dentro da árvore. [GKNG03] A partir das DOM trees, modelos de estruturação para esses dados podem ser construídos e assim o pre-processamento deles se transforma no de dados estruturados.

Outra abordagem, que foi a utilizada no projeto, é a de extração textual completa da página. Ou seja, são indentificadas as tags HTML, que são ignoradas, deixando assim somente o texto cru contido em cada página. O texto então é tratamento de forma a transformar cada palavra contida na página em um atributo do conjunto de treinamento. Esses atributos passam por processos de rankeamento e relevancia que serão melhor descritos na próxima subseção.

2.1.2.2 Técnicas para tratamento de texto

As técnicas de tratamento de texto que iremos abordar abragem o campo de tokenização e de seleção de atributos.

A tokenização é um processo de análise do texto pre-processado ou transformado em texto cru para que a partir dele sejam definidas palavras. Esse processo vai definir dentro do texto o que é uma palavra e em qual classe sintática aquela palavra pertence. [GG94] O texto é tratado como uma string única que será divida em pequenas unidades e cada uma delas será interpretada e designada um classe sintática. Além disso, a tokenização também define o que são sentenças e onde elas começam e terminam, verificando pontuações no texto.

Após a tokenização, a seleção de atributos vai filtrar os tokens do texto para selecionar atributos a partir deles. O processo de seleção consiste em filtrar tokens que representam palavras de pouco uso e muito uso. Filtrando assim tokens de nomes próprios e preposições. E após esse processo há a seleção de tokens que serão classificados como atributos para o modelo. Existem várias abordagens para rankear a importancia desse atributos sendo as mais conhecimento tf (term frequency) e idf(inverse document frequency), que muitas vezes são usados combinadamente.

A frequencia do termo ou TF, simplesmente faz uma contagem de quantas vezes aquele termo (ou token) aparece em um documento (ou texto). Já o inverso da frequencia no documento é incorporado ao tf (por isso são muitas vezes utilizados em conjunto) para diminuir a importância de termos que tem uma frequencia muito grande. [LPJ02]

Após a seleção de atributos e o rankeamento da importancia das palavras no documento ele pode ser utilizado como conjunto de dados para um modelo de aprendizado supervisionado. Porém, outros ajustes como limitação de quantidade de atributos podem ser feitos para melhorar o desempenho do conjunto.

2.2 Aprendizado supervisionado

Aprendizado supervisionado é uma tecnica de aprendizagem de máquina que consiste em criar um modelo com uma função que mapea uma entrada à uma saída, de acordo com um conjunto de entradas e saídas já conhecido. Esse conjunto é chamado de conjunto de treinamento. O conjunto de treinamento é analisado pelo algoritmo que cria uma função que irá mapear novas entradas para saídas corretas, de acordo com ele.

Ou seja, no aprendizado supervisionado, para cada entrada do conjunto de treinamento $x_i, i = 1, ..., n$ existe uma medida associada y_i que é avaliada pelo algoritmo. Assim, um modelo que se encaixe nas condições dadas pelo conjunto de treinamento é criado com o intuito de prever valores de futuras observações ou de entender melhor o relacionamento entre entradas e saídas do algoritmo. [Kot07]

Há dois tipos de modelos de aprendizado supervisionado, sendo esses o modelo de classificação e o de regressão. Geralmente associamos os problemas quantitativos a regressão, enquanto os qualitativos são associados a classificação. Porém, a diferença entre os modelos não é tão simples assim. [Kot07] O modelo de classificação tem como objetivo criar uma função que, a partir de um conjunto de treinamento rotulado, consiga definir a que classe uma nova

entrada pertence. Nesse caso, o conjunto de atributos dos dados de treinamento são avaliados e sua classe (ou rótulo) é verificada. A função aprende que características levar em conta quando vai avaliar uma nova entrada para assim rotulá-la. [Spi94] O modelo de regressão tem um objetivo parecido com o de classificação, porém a função criada pelo modelo tem que conseguir prever um valor associado a uma entrada. No caso, ao invés de tentar prever um classe a partir dos atributos do conjunto de treinamento, o algoritmo tenta prever um possível valor para novas entradas a partir desses atributos. A função aprende a partir do conjunto de treinamento que valor deve atribuir para cada nova entrada, de acordo com os valores presentes no conjunto. Para ambos os casos, após o processo de construção e aperfeiçoamento da função do modelo, chamado de treinamento, a função é validada a partir de um conjunto de testes. Essa validação pode ser verificada a partir das métricas de avaliação.

Nesta seção, iremos abordar alguns algoritmos de aprendizado supervisionado, focando nos que envolvem regressão, e suas métricas de avaliação.

2.2.1 Algoritmos de aprendizado supervisionado

Nesta subseção iremos abordar alguns algoritmos de aprendizado supervisionado especificamente ligados a regressão. Todos os algoritmos foram utilizados nesse trabalho.

2.2.1.1 Random Forest

O algoritmo de Random Forest é um algoritmo baseado em um outro método chamado Decision Tree (ou Árvore de Decisão). Árvore de Decisão é uma algoritmo que utiliza uma estratificação do conjunto de dados em formato de árvore. É uma forma de mapeamento de decisões em que cada folha da árvore é um possível resultado da regressão. Ou seja, de acordo com uma serie de valores escolhidos para os atributos, temos um resultado final: a folha da árvore. Portanto, várias possibilidades de valores são avaliados para os atributos e o valor de cada folha representa o resultado das escolhas dos valores anteriores.

O Random Forest é classificado como um algoritmo de "aprendizado em conjunto"ou ensemble learning, portanto utiliza várias Árvores de Decisão em conjunto com uma seleção aleatória de partes do conjunto de dados de treinamento para compor essas árvores. Para cada árvore, um subconjunto de treinamento aleatório com uma seleção de atributos aleatória (também seccionando os atributos dos dados) é criado e executado e ao fim uma média dos resultados é feita para cobrir os valores que não foram executados. Além disso, o Random Forest consegue avaliar a relevancia dos atributos sobre o conjunto de treinamento, dado que as árvores que tiverem os atributos mais relevantes terão resultados melhores e o algoritmo pode ir se ajustando para selecionar esses atributos para a avaliação do conjunto de testes.

2.2.1.2 Support Vector Machine - Regression

O Máquina de Vetor Suporte, Support Vector Machine ou SVM é um algoritmo muito usado para classificação e também tem uma versão conhecida para regressão o SVR (Support Vector Regression). O SVM divide o espaço do conjunto de dados através vetores de suporte de forma a deixar os pontos de classes diferentes com uma distancia considerável. O vetor que tem a

maior margem é escolhido entre os demais para definir as classes. A margem é definida pelo ponto do espaço que está mais perto da reta. Portanto, os pontos mais distantes da margem teriam sua classe mais fortemente definida, enquanto os que estão mais próximos não tem uma definição tão precisa.

O SVR funciona similarmente ao SVM, só que ao invés de classificar os pontos do espaço, queremos estimar um valor numérico. Os vetores suporte são definidos no espaço do conjunto de treinamento de forma a tentar estimar os valores ideiais para os resultados. [Bha18] Após isso uma margem (epsilon) é adicionada ao vetor. Essa margem funciona como uma margem de erro para o valor esperado definido pelo vetor. Portanto os pontos que serão levados em consideração serão os pontos que estarão dentro da margem de erro.

2.2.1.3 XGBoost

O XGBoost ou eXtreme Gradient Boosting é uma implementação de árvores de decisão de aumento de gradiente com foco em rapidez e perfomace. [Bro16] O XGBoost está sendo fortemente usado em competições e desafios de aprendizado de máquina e big data por sua rapidez. A implementação do algoritmo foi feita para ter a maior eficiência possível no uso de recursos de memória e processamento. Assim tendo as melhores condições possíveis para o treinamento de um modelo.

2.2.2 Métricas de avaliação de um modelo de regressão

Nesta subseção iremos falar sobre algumas métricas de avaliação dos algoritmos de regressão já descritos. Todas essas métricas foram utilizadas neste trabalho.

Iremos definir para as equações seguintes $y =$ valor real e $\hat{y} =$ valor esperado.

2.2.2.1 Erro médio quadrado (MSE)

O erro médio quadrado calcula a média dos erros do modelo ao quadrado, com isso, os valores maiores terão mais importância do que os menores. Já que serão levados em consideração os valores ao quadrado. O calculo pode ser descrito pela seguinte equação:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.1}$$

2.2.2.2 Erro médio absoluto (MAPE)

O erro médio absoluto calcula a porcentagem da média dos erros do modelo em valor absoluto, portanto é aplicada uma modulação à subtração. O peso de importância aos erros é dados de maneira linear. O calculo pode ser descrito pela seguinte equação:

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{2.2}$$

2.2.2.3 Erro médio do logaritmo da diferença (MLE)

O erro médio do logaritmo da diferença calcula a média dos logaritmos dos erros. O propósito do uso é para não penalizar discrepancias muito grandes em valores de erros no conjunto. O calculo pode ser representado pela seguinte equação:

$$MLE = \frac{1}{n}\sum_{i=1}^{n}((\log y_i) - (\log \hat{y}_i)) \tag{2.3}$$

2.2.2.4 Coeficiente de determinação (R2)

O coeficiente de determinação é uma métrica de avaliação de regressão que mede quão perto o valor esperado está de se encaixar no modelo. E pode ser descrito como a porcentagem de variação da variavel resultante do modelo ou mesmo:

$$R2 = \frac{V_r}{V_t} \tag{2.4}$$

Onde $V_r =$ Variação da variavel resultante e $V_t =$ Variação total.

CAPÍTULO 3

Implementação do Regressor

Como mencionado anteriormente, esse trabalho tem como objetivo construir um regressor que faça a predição de preços de imóveis a partir de dados estruturados e dados textuais, não-estruturados, para que seja possível fazer uma comparação do desempenho entre esses tipos de dados, como também em relação a combinação dos dois tipos. Ou seja, podermos ao fim definir qual o impacto dos dados textuais na definição do preço do imóvel pelo regressor.

Esse tipo de avaliação de dados textuais na predição de preços tem sido uma tendencia que vem crescente nos últimos anos. Como já citamos anteriomente, a Kaggle [Mer18] em conjunto com outras empresas vem fazendo desafios e competições que abordam esses temas.

Neste capítulo iremos descrever o processo utilizado para a implementação desse regressor. Descrevendo como os dados foram recebidos e pre-processados, que algoritmos de regressão e técnicas de apredizagem de máquina foram utilizados, como também as dificuldades encontradas na implementação. A figura 3.1 descreve o pipeline da implementação:

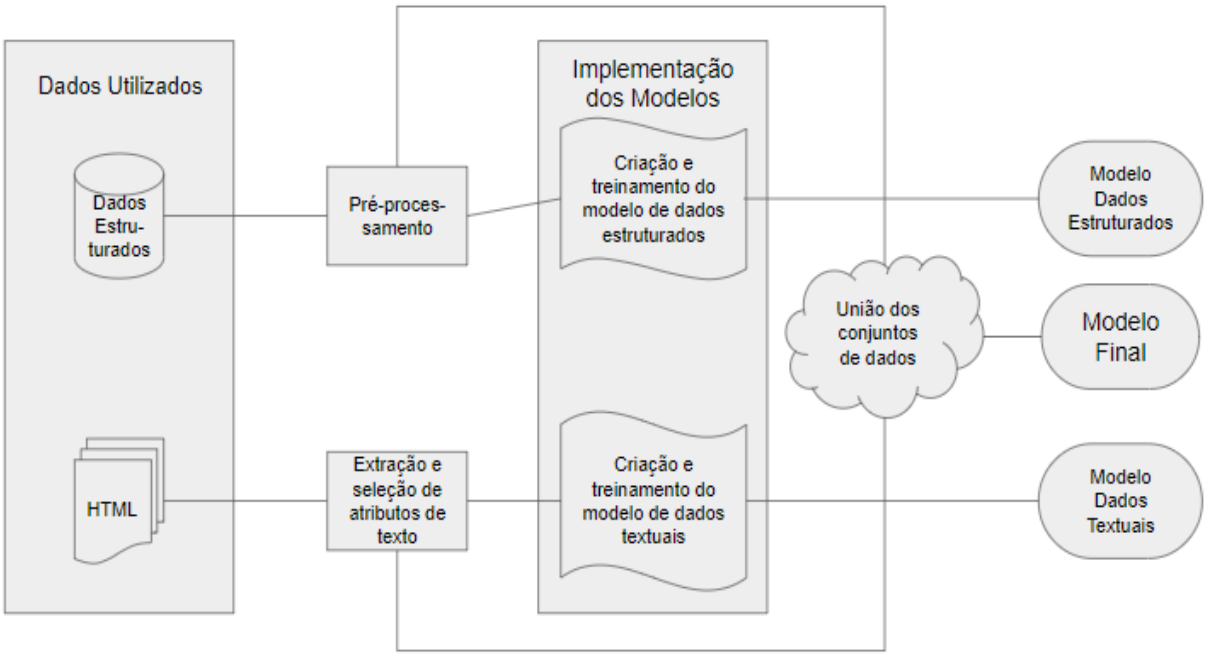

Figura 3.1 Pipeline da implementação do algoritmo

3.1 Dados Utilizados e Pré-processamento

3.1.1 Dados Utilizados

Os dados utilizados foram obtidos a partir de um coletor focado no domínio de imóveis, que coleta informações de varios dados da web brasileira. Foram escolhidos a partir da extração, dados de sites de imóveis de 3 cidades brasileiras: Rio de Janeiro (RJ), São Paulo (SP) e Porto Alegre (POA). Para cada uma dessas cidades foram extraídos cerca de dez mil imóveis para compor o conjunto de dados do experimento. A tabela 3.1 descreve a distribuição dos dados.

Status	RJ	SP	POA
Antes do Pré Processamento	7128	10000	10000
Após o Pré Processamento	5208	7304	9489

Tabela 3.1 Quantidade de instancias por Cidade

Os dados estruturados foram recebidos já no formato csv com 8 colunas. Sendo essas colunas: "price"representando o preço, "latitude", "longitude", "bedrooms"representando a quantidade de quartos no imóvel, "bathrooms"representando a quantidade de banheiros no imóvel, "area"representando a área em m², "pkspaces"representando a quantidade de vagas de estacionamento do imóvel, "ensuites"representando a quantidade de suítes presentes no imóvel, "timestamp"representando o horário ao qual o site foi visitado, "type"representando o tipo do imóvel, sendo casa ou apartamento, "operation"representando a operação a qual o imóvel estava sendo ofertado, sendo venda ou aluguel e "url"representando a url do site no qual os dados do imóvel foram extraídos.

Os dados não-estruturados foram recebidos como as páginas HTML presentes na coluna "url"dos dados estruturados. Todas as páginas foram baixadas e adicionadas ao conjunto de dados não-estruturados. Ao fim iremos descrever também como foi a união desses dados estruturados e não-estruturados.

O tratamento desses dados e as tecnologias utilizadas serão descritas nas próximas subseções. Todas a implementação do regressor foi feita em Python e suas bibliotecas adicionais.

3.1.2 Pré-processamento

Os dados estruturados em formato .csv foram carregados em memória utilizando a biblioteca de python Pandas. [r1618] Essa biblioteca transforma arquivos csv em Data Frames. Esses Data Frames funcionam como tabelas, onde os dados podem ser acessados através das colunas rotuladas anteriormente.

Após a construção do Data Frame, foi feita uma análise dos dados presentes e foi visto que para o regressor colunas como a de "url"e "timestamp"não eram interessantes para o algoritmo. Portanto, essas colunas foram removidas. Além disso algumas colunas do Data Frame possuiam strings que descreviam o atributo. Essas colunas eram "operation"e "type"que tinha como únicas entradas de string, respectivamente, "sell"e "rent", descrevendo venda ou aluguel,

e "house"e "apartment", descrevendo o tipo do imóvel como casa ou apartamento. Essas colunas foram categorizadas e agora ficaram representadas pelas valores 1 e 0 ao invés das strings.

Foi feita também uma verificação para valores nulos no Data Frame e as colunas e linhas com valores nulos foram removidos. Percebeu-se também que os imóveis (ou linhas) que possuíam valor de "operation"igual a "rent", ou seja imóveis que estavam sendo ofertados como aluguel, estavam com valores de preço muito baixos. Portanto, foram classificados como outliers e foram removidos do Data Frame. Isso pode ser observado no exemplo do RJ nas Figuras 3.2 e 3.3 a seguir:

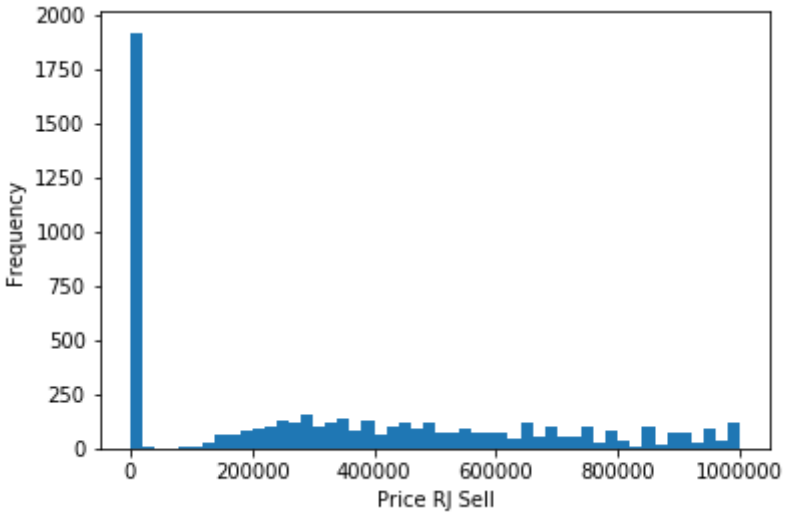

Figura 3.2 Gráfico de Preço com Aluguel

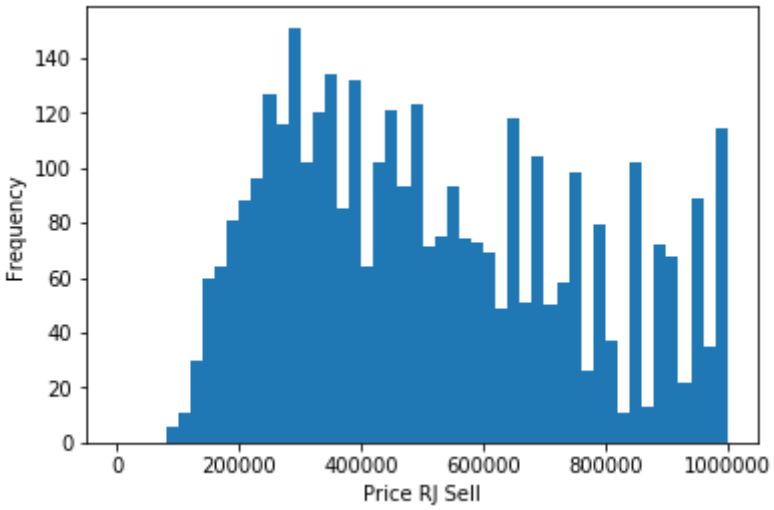

Figura 3.3 Gráfico de Preço Somente com Venda

Após o tratamento dos dados e remoção de entradas indesejadas, foi feita a separação dos

dados em conjuntos de treinamento e teste, onde foi dividido uma porcentagem de 80 (oitenta) por cento para treinamento e 20 (vinte) para teste. Assim, os dados de treinamento e teste já tratados foram gravados em novos arquivos csv para serem utilizados pelo script de criação do modelo do regressor.

3.1.3 Extração e seleção de atributos de texto

Os dados não-estruturados foram recebidos em formato HTML, então foi utilizado uma biblioteca de Python chamada Beautiful Soup [r1718] para extrair todo o texto contido nas páginas html e assim grava-los em novos arquivos txt.

Antes do processo de extração e criação dos arquivos txt, os arquivos HTML foram ordenados para que os novos arquivos txt tivessem o mesmo index do imóvel indicado nos Data Frames já criados. Esse processo facilitaria a integração dos dados estruturados e não-estruturados no futuro. Além disso, os arquivos txt foram separados em duas pastas diferentes que diferenciavam treinamento e teste, também de acordo com os data frames.

Após a criação dos arquivos txt, foi utilizada uma pacote da bibliteoca Sci-Kit Learn [r1818], também utilizadas na criação de modelos, chamado TfidfVectorizer [r1918], para o carregamento dos arquivos txt em dois vetores. Uma para o conjunto de treinamento e outro para o conjunto de teste. Esses vetores são utilizados para realizar o processo de seleção de atributos, já citado no capítulo 2, onde é delimitado uma quantidade máxima de features e também um arquivo que contenha stopwords, que são palavras de uso frequente que devem ser ignoradas pelo vetor na hora do calculo de relevância das palavras. A quantidade de atributos escolhida inicialmente para os vetores foi de 18000 (dezoito mil) atributos, porém por conta de alguns problemas, que serão discutidos em outra seção mais a frente, esse valor foi reduzido para 1000 (mil) atributos.

3.2 Implementação dos modelos

Os modelos de regressão foram criados com a utilização da biblioteca Sci-kit Learn [r1818]. Essa biblioteca fornece vários regressores com algoritmos de aprendizado de máquina diferentes. Os algoritmos escolhidos foram: Random Forest, SVR e XGBoost.

Inicialmente foram construídos dois modelos, um que recebia e utilizava os dados estruturados e outro que utilizava os não estruturados. Após o pré-processamento dos dados, rodar os regressores sobre esses dados utilizando o sci-kit learn foi bem simples. Os modelo foram construídos a partir do conjunto de treinamento e após isso foram testados sobre o conjunto de testes. O teste foi medido a partir das métricas citadas no capítulo 2. Os resultados e comparações sobre esses resultados serão analizados no capítulo 4.

Após as duas versões iniciais, foi construído um novo modelo que unia os conjuntos de entrada de dados estruturados e não-estruturados. Iremos nos aprofundar nessa construção nas seções seguintes.

Primeiramente, foi implementado um regressor utilizando o algoritmo Random Forest, já citado no capítulo 2. O pacote utilizado foi o ensemble do Sci-kit Learn [r1818]. Os parâmetros do regressor foram definidos a partir do SMAC. O SMAC foi utilizado para otimizar o regres-

sor ajustando os valores dos parâmetros para um melhor resultado do algoritmo. O SMAC foi somente usado para o Random Forest, pois tinha uma implementação mais simples e foi relevante para um levantamento inicial. Porém, aumentou muito o tempo de execução da criação do modelo. Isso resultou na opção de não utilizá-lo para os demais algoritmos de classificação.

Além do algoritmo de Random Forest, também foram utilizados regressores com os algoritmos SVR e XGBoost. Os parâmetros utilizados para o SVR foram os valores padrão definidos pelo Sci-kit Learn [r1818]. Foi utilizado para a implementação desse regressor o pacote SVM do Sci-kit learn. Já para o XGBoost, foi utilizado o pacote de Python xgboost e os parâmetros utilizados foram o padrão definido pelo pacote.

3.3 União dos conjuntos de dados

Nesta seção iremos discutir como foi implementado o regressor que utilizava os dois conjuntos de dados, estruturado e não-estruturado.

O regressor teve uma implementação bem parecida com o dos dados estruturados. O Data Frame em pandas foi mantido, assim como os TfidfVectorizer. Porém, o vetor foi convertido em um novo Data Frame onde cada coluna representava uma feature do texto. Nas linhas desse Data Frame estavam os valores de TFIDF de cada uma das features e quando não houvesse valor para aquela feature em um certo documento o valor NaN era atribuído.

Após a construção desse novo Data Frame a partir do vetor foi feita a união dos dois Data Frames. Assim temos um novo Data Frame que possui os atributos estruturados e as features do texto como colunas. Após a união os valores NaN foram substituídos por 0.0, para assim não terem relevancia no calculo do regressor, mas serem contabilizados como dados.

Como dito na seção anterior, a limitação de features foi essencial para que o desempenho do algoritmo do regressor da união desses dados fossem bom. Inicialmente foram definidas 10000 (dez mil) features para texto, porém o mesmo problema, que havia acontecido com POA anteriormente, aconteceu para os outros conjuntos de dados. Então, assim como para os dados não-estruturados, foi definido um limite de 1000 (mil) features para texto.

O algoritmo executou bem para 1000 features, porém ainda teve um tempo de execução muito alto. Durou alguns dias para ser executado, mesmo para o conjunto do RJ que tinha uma quantidade de entradas bem menor (devido a remoção de imóveis para alugar).

3.4 Dificuldades e problemas encontrados

Nesta seção iremos detalhar alguns dos problemas encontrados durante a implementação do regressor, muitos dos problemas foram relacionados a estouro de memória e tempo de execução dos algoritmos. As próximas subseções irão detalhar também as soluções para estes problemas e o impacto deles na implementação final do algoritmo.

Inicialmente, durante a implementação do regressor utilizando dados textuais, não foi configurado um limite para a quantidade de features (ou atributos). Além disso, o vetor escolhido para definir a relevancia foi o CountVectorizer, que define a relevancia somente pela quantidade de repetições no documento. Isso impactou em uma quantidade de features muito grande, onde

o maior número, que foi o de POA, tinha mais de 45000 (quarenta e cinco mil) features.

Essa quantidade grande de features teve um impacto no desempenho do algoritmo. Quando executado demorou mais de 1 dia para terminar, e no caso de POA foi interrompido por um erro de estouro de memória. Então, foi decidido limitar a quantidade de features para 1800 (dezoito mil). Além disso, também foi preferível mudar o vetor sendo utilizado. Ou invés de utilizar o CountVectorizer, foi escolhido o TfidfVectorizer, que como dito anteriormente, conseguia definir melhor a relevância das palavras.

Após essas duas mudanças, os algoritmos ficaram muito mais rápidos, porém como a quantidade de dados textuais no conjunto de POA era maior do que os de SP e RJ, ainda assim não foi possível finalizar a execução. O erro de estouro de memória ainda persistiu.

Então, foi decidido deixar esse conjunto de dados de lado e trabalhar somente com os outros dois menores. Por questões de limitações da máquina, já que a quantidade de features continuava muito alta para o conjunto de POA.

Porém, quando implementada a união dos conjuntos de dados, que irá ser detalhada na próxima seção, problemas semelhantes surgiram. Portanto, foi decidido limitar mais ainda o número de features para somente 1000 (mil). Essa mudança foi essencial para a execução do conjunto de dados de POA e não impactou tanto nos resultados que serão analisados no capítulo 4.

CAPÍTULO 4

Análise dos Resultados

Neste capítulo iremos analisar a configuração do ambiente e como os conjuntos de treinamento e teste de cada cidade foram divididos e também os resultados gerados por cada um dos regressores implementados.

4.1 Configuração do ambiente

Como já discutido no capítulo 3, os conjuntos de dados de cada cidade foram divididos em treinamento e teste para que fossem implementados os algoritmos de regressão. A divisão treinamento/teste foi dada por uma porcetagem de 80 por cento do conjunto de dados para treinamento e 20 por cento para teste.

A divisão dos dados para todas as cidades estão descritos pela tabela 4.1.

Conjunto	RJ	SP	POA
Treinamento	4199	5875	7595
Teste	1009	1429	1894

Tabela 4.1 Quantidade de instancias por Cidade nos conjuntos de treinamento e teste

Os regressores utilizados para predizer o preço dos imóveis foram: Random Forest, SVR e XGBoost. Para os dados não-estruturados foram utilizados vetores TFIDF, com 1000 atributos para a regressão. Quando foi feita a união das duas entradas, um novo data frame foi gerado sendo a união do data frame dos dados estruturados, já pré-processados, e de um data frame gerado a partir do vetor de 1000 atributos. Gerando assim um data frame com as colunas originais já citadas e mais 1000 novas colunas para cada atributo extraído do pré-processamento textual.

4.2 Resultados

Nesta seção iremos mostrar e interpretar os resultados de cada um dos classificadores criados. Nas circunstâncias definidas na configuração do ambiente, e para cada um dos tipos de dados e sua combinação. Sendo possível assim uma comparação de cada um dos regressores a partir das métricas utilizadas, já citadas no capítulo 2.

4.2.1 Dados Estruturados

Os resultados do regressor Random Forest de dados estruturados estão demonstrados pela tabela 4.2.

Métrica	RJ	SP	POA
MSE	2.1501×10^{11}	2.7416×10^{11}	9.4513×10^{10}
MAPE	24.3408	30.5298	33.6817
MLE	0.2570	0.3216	0.3395
R2	0.6804	0.4680	0.3766

Tabela 4.2 Resultados Random Forest Dados Estruturados

Podemos ver os resultados do regressor SVR de dados estruturados através da tabela 4.3.

Métrica	RJ	SP	POA
MSE	9.8832×10^{11}	8.6975×10^{11}	2.6728 x 10^{11}
MAPE	89.2273	79.3547	72.4214
MLE	0.6807	0.5880	0.5621
R2	-7.2925	0.0	-1.9722

Tabela 4.3 Resultados SVR Dados Estruturados

A tabela 4.4 descreve os resultados do regressor XGBoost para os dados estruturados.

Métrica	RJ	SP	POA
MSE	2.1458 x 10^{11}	1.7106 x 10^{11}	5.8387 x 10^{10}
MAPE	25.2233	22.7950	25.5003
MLE	0.2436	0.2367	0.2570
R2	0.6788	0.6984	0.6834

Tabela 4.4 Resultados XGBoost Dados Estruturados

A partir dos dados descritos nas tabelas, podemos observar que o regressor XGBoost obteve os melhores resultados para os conjuntos de dados de todas as cidades. Tendo valores de erro menores.

4.2.2 Dados Não-Estruturados

Os resultados dos dados não-estruturados para o regressor Random Forest pode ser visualizado a partir da tabela 4.5.

O regressor SVR para dados não-estruturados apresentou os resultados descritos pela tabela 4.6.

Os dados não-estruturados apresentaram os resultados descritos pela tabela 4.7 para o regressor XGBoost.

Métrica	RJ	SP	POA
MSE	8.9703 x 10^{11}	7.9540 x 10^{11}	2.4165 x 10^{11}
MAPE	66.2063	62.2558	57.9385
MLE	0.7447	0.6664	0.5910
R2	-63.1374	-107.1170	-107.1295

Tabela 4.5 Resultados Random Forest Dados Não-Estruturados

Métrica	RJ	SP	POA
MSE	9.8522 x 10^{11}	8.7004 x 10^{11}	2.6762 x 10^{11}
MAPE	88.5768	79.4119	72.6002
MLE	0.6807	0.5880	0.5622
R2	-1327337390969308.8	-3505266873730457.0	-946071832225996.8

Tabela 4.6 Resultados SVR Dados Não-Estruturados

Métrica	RJ	SP	POA
MSE	9.2185 x 10^{11}	8.1141 x 10^{11}	2.4528 x 10^{11}
MAPE	66.7736	62.8796	58.0812
MLE	0.7488	0.6668	0.5943
R2	-32.5899	-44.3245	-42.1948

Tabela 4.7 Resultados XGBoost Dados Não-Estruturados

A partir das tabelas apresentadas, podemos inferir que os resultados dos dados de texto foram bem piores do que os dados estruturados. Tendo o conjunto com mais instâncias, POA, como o melhor dentre os outros. Com variações de conjunto para conjunto em cada um dos algoritmos.

4.2.3 Combinação dos Dados

Apesar dos resultados terem sido piores para os dados textuais, quisemos verificar como a combinação dos dados textuais iriam interferir nos resultados dos dados estruturados, e se seria possível uma melhora nos resultados.

A tabela 4.8 descreve os resultados do regressor Random Forest para a combinação dos dados estruturados e não-estruturados.

Os resultados do regressor SVR para a combinação dos dados estruturados e não-estruturados podem ser visualizados na tabela 4.9

A combinação dos dados estruturados e não-estruturados apresentaram os resultados descritos pela tabela 4.10 para o regressor XGBoost.

Podemos observar nas tabelas apresentadas que obtivemos uma melhora não muito significativa em algumas métricas para os regressores apresentados nas Tabelas 4.8, 4.9 e 4.10. Iremos analisar mais afundo como foram essas mudanças apresentando os gráficos contidos nas Figu-

Métrica	RJ	SP	POA
MSE	1.7831 x 10^{11}	2.8138 x 10^{11}	6.4852 x 10^{10}
MAPE	23.2393	31.4526	26.4236
MLE	0.2444	0.3328	0.2661
R2	0.7446	0.4133	0.6301

Tabela 4.8 Resultados Random Forest Dados Combinados

Métrica	RJ	SP	POA
MSE	9.0878 x 10^{11}	7.7300 x 10^{11}	2.6268 x 10^{11}
MAPE	87.98	75.8377	71.9970
MLE	0.6871	0.5746	0.5615
R2	-4.6710	-1.5121	-7928168721069216.0

Tabela 4.9 Resultados SVR Dados Combinados

Métrica	RJ	SP	POA
MSE	1.9145 x 10^{11}	1.7747 x 10^{11}	6.2604 x 10^{10}
MAPE	24.7457	23.8949	26.1248
MLE	0.2626	0.2534	0.2653
R2	0.7026	0.6375	0.6476

Tabela 4.10 Resultados XGBoost Dados Combinados

ras 4.1, 4.2 e 4.3, para comparar os resultados obtidos com cada um dos tipos de entrada de dados.

Podemos ver a partir dos gráficos, que os resultados dos dados estruturados e combinados tem valores semelhantes ou próximos. O que nos indica que de o impacto dos dados não-estruturados na regressão foi bastante baixo. Podemos ver também a discrepância dos valores para somente os dados não-estruturados em relação aos outros tipos. Assim como, também podemos ver que o regressor SVR teve desempenho bem abaixo dos demais.

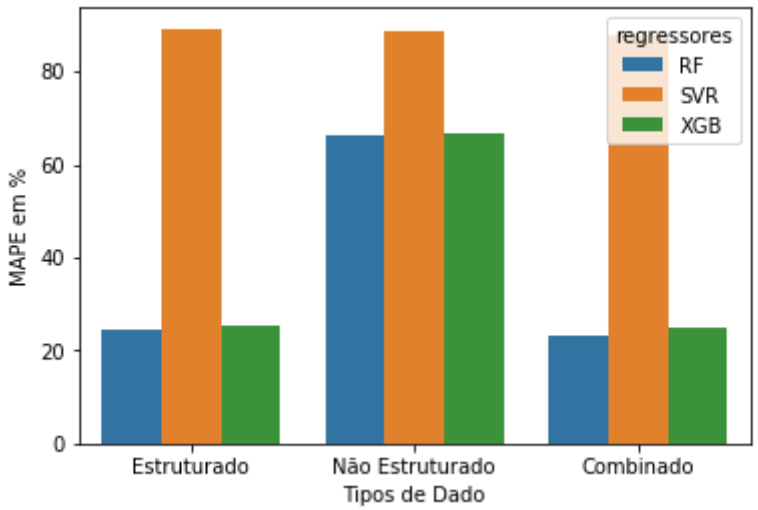

Figura 4.1 Comparação tipos de dado x MAPE para RJ

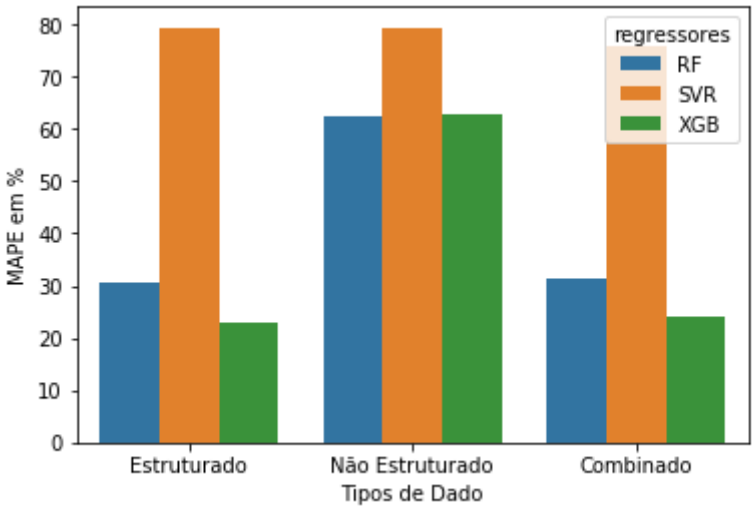

Figura 4.2 Comparação tipos de dado x MAPE para SP

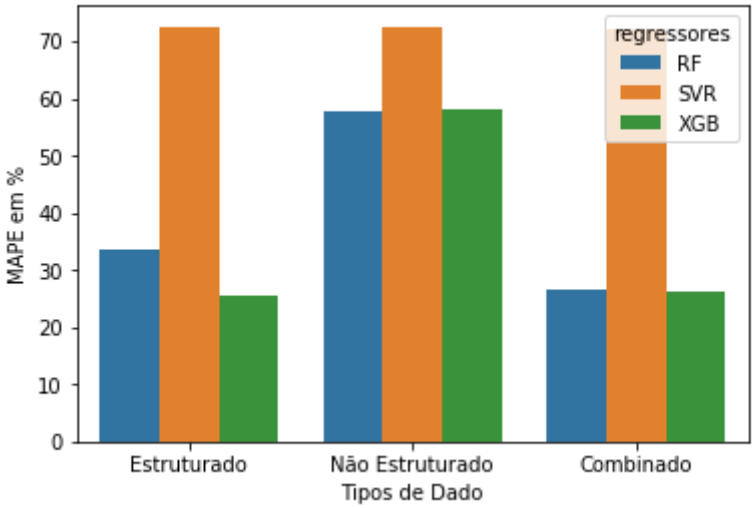

Figura 4.3 Comparação tipos de dado x MAPE para POA

CAPÍTULO 5

Conclusão

Neste trabalho foram apresentados vários regressores com o intuito de prever o preço de imóveis, de acordo com os conjuntos de dados de 3 grandes cidades brasileiras, como também de comparar os resultados dos dados estruturados e não-estruturados nessa predição e qual o impacto dos dados não-estruturados quando a combinação deles é utilizada. Além disso, foi apresentado o pré-processamento tanto dos dados estruturados, quanto dos dados não-estruturados. Realizando processos como processamento de texto e análise de data frames.

Os algoritmos de regressão utilizados tiveram resultados bons para os dados não estruturados, tendo como o melhor deles o XGBoost ficando na frente de Random Forest e SVR, nesta ordem. Para os dados não-estruturados, notamos que os conjuntos que tinham mais instâncias tiveram resultados melhores para todos os algoritmos. Como o conjunto com mais instancias era POA, tivemos o melhor resultado desse conjunto no algoritmo de Random Forest.

Para a combinação dos dados estruturados tivemos resultados que não foram muito expressivos em relação aos que tinham sido obtidos somente com os dados estruturados. Isso indica que o impacto dos dados não-estruturados sobre os estruturados na definição do preço dos imóveis é mínima. Tivemos como os melhores regressores o XGBoost e o Random Forest, que tiveram resultados bem próximos. Já o SVR teve um desempenho muito ruim, sendo até distoante dos demais regressores.

Como a quantidade de features geradas pelos vetores de texto foi muito grande, limitou-se as features do vetor dessa forma obteve-se menos informação sobre os dados textuais, o que pode ter prejudicado o desempenho dos regressores que utilizaram dados textuais. Porém, a limitação de features foi necessária para que o algoritmo conseguisse rodar em tempo hábil. Tanto quando utilizou somente os dados textuais, e principalmente quando utilizou a combinação dos dois. O problema da quantidade de features se agrava para a combinação pois a entrada se torna um data frame com uma quantidade de colunas igual a de features mais 8, que eram as que já existiam inicialmente. Isso torna o conjunto de dados com uma quantidade muito grande de colunas, o que resulta em tempo de execução maior.

Para estudos futuros, seria possível avaliar quais são essas features que estão sendo escolhidas e se a relevancia aplicada a ela está sendo aplicada corretamente pelo algoritmo. Além disso, também seria interessante conduzir um estudo que avaliasse a partir consumidores que atributos presentes, tanto nos dados estruturados quanto nos textos de descrições, avaliados nos dados não-estruturados, seriam relevantes para eles na valorização de um imóvel. Isso traria um novo viés de rankeamento que poderia definir pesos mais precisos para a relevancia desses atributos, o que, por fim, também aumentaria a precisão do algoritmo em relação a previsão de preço.

Referências Bibliográficas

[ABT16] Eric Medvet Alberto Bartoli, Andrea De Lorenzo and Fabiano Tarlao. Inference of regular expressions for text extraction from examples. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, 28(5), May 2016.

[Bha18] Indresh Bhattacharyya. Support vector regression or svr. *Medium*, June 2018.

[Bro16] Jason Brownlee. A gentle introduction to xgboost for applied machine learning. August 2016.

[DC10] Sylvain Chartier Denis Cousineau. Outliers detection and treatment: a review. *International Journal of Psychological Research*, 3(1), 2010.

[Fre18] Freiburg. Auto ml. 2018.

[GG94] Pasi Tapanainen Gregory Grefenstette. What is a word, what is a sentence? problems of tokenization. *Rank Xerox Research Centre Grenoble Laboratory*, April 1994.

[GJ13] Trevor Hastie Robert Tibshirani Gareth James, Daniela Witten. *An Introduction to Statistical Learning with Applications in R*. 2013.

[GKNG03] Suhit Gupta, Gail Kaiser, David Neistadt, and Peter Grimm. Dom-based content extraction of html documents. *Proceedings of the 12th International Conference on World Wide Web*, pages 207–214, 2003.

[JVdB05] Roger Eeckels Kobus Herbst Jan Van den Broeck, Solveig Argeseanu Cunningham. Data cleaning: Detecting, diagnosing, and editing data abnormalities. September 2005.

[Kot07] S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, (22):3–24, 2007.

[LPJ02] Hong-Bo Shi Li-Ping Jing, Hou-Kuan Huang. Improved feature selection approach tfidf in text mining. *Proceedings. International Conference on Machine Learning and Cybernetics*, November 2002.

[Mer18] Mercari. Price suggestion challenge: Can you automatically suggest product prices to online sellers? 2018.

[Pyl99] Dorian Pyle. *Data Preparation for Data Mining*. 1999.

[QY17] Liangliang Cao Jiebo Luo Quanzeng You, Ran Pang. Image-based appraisal of real estate properties. *IEEE Transactions on Multimedia*, 19(12):2751 – 2759, December 2017.

[r1618] Python data analysis library. 2018.

[r1718] Beautiful soup. 2018.

[r1818] Sci-kit learn. 2018.

[r1918] Tfidfvectorizer. 2018.

[Spi94] Susan Spiggle. Analysis and interpretation of qualitative data in consumer research. *Journal of Consumer Research*, 21(3):491–503, December 1994.

Este volume foi tipografado em LaTeX na classe UFPEThesis (www.cin.ufpe.br/~paguso/ufpethesis).