UNIVERSIDADE DE TAUBATÉ

Leonardo Geraldo Alves Ferreira

ESTUDO SOBRE BIBLIOTECAS DE INTELIGÊNCIA ARTIFICIAL E APRENDIZADO DE MÁQUINA PARA A LINGUAGEM DE PROGRAMAÇÃO PYTHON

Taubaté - SP

2023

Grupo Especial de Tratamento da Informação - GETI
Sistema Integrado de Bibliotecas – SIBi
Universidade de Taubaté - Unitau

F383e Ferreira, Leonardo Geraldo Alves
Estudo sobre bibliotecas de inteligência artificial e aprendizado de máquina para a linguagem de programação python / Leonardo Geraldo Alves Ferreira. -- 2023.
65 f. : il.

Monografia (graduação) – Universidade de Taubaté, Departamento de Informática, 2022.
Orientação: Prof. Dr. Luis Fernando de Almeida, Departamento de Informática.

1. Python. 2. Inteligência Artificial. 3. Aprendizado de máquina. 4. Software. I. Universidade de Taubaté. Departamento de Informática. Graduação em Engenharia da Computação. II. título.

Ficha catalográfica elaborada **por Aline Peregrino – CRB-8/10970**

Leonardo Geraldo Alves Ferreira

ESTUDO SOBRE BIBLIOTECAS DE INTELIGÊNCIA ARTIFICIAL E APRENDIZADO DE MÁQUINA PARA A LINGUAGEM DE PROGRAMAÇÃO PYTHON

Trabalho de Conclusão de Curso apresentado como requisito parcial para a obtenção do diploma de Bacharel em Engenharia de Computação pela Universidade de Taubaté.

Área: Inteligência Artificial

Orientador: Prof. Dr. Luis Fernando de Almeida

Taubaté - SP

2023

LEONARDO GERALDO ALVES FERREIRA

Estudo sobre bibliotecas de inteligência artificial e aprendizado de máquina para a linguagem de programação Python

Trabalho de Conclusão de Curso apresentado como requisito parcial para a obtenção do diploma de Bacharel em Engenharia de Computação pela Universidade de Taubaté.

Área: Inteligência Artificial

Data:_____

Resultado:_____

BANCA EXAMINADORA

Prof. Dr. Luis Fernando de Almeida — Universidade de Taubaté

Assinatura:_____

Prof. Me. Luiz Eduardo de Souza Evangelista — Universidade de Taubaté

Assinatura:_____

Prof. Me. Fabio Rosindo Daher de Barros — Universidade de Taubaté

Assinatura:_____

AGRADECIMENTOS

Agradeço a todos que sempre acreditaram em mim, e com profunda gratidão expresso meu carinho e afeto a todos que compartilharam da minha jornada. Por mais esforço e dedicação que gastemos em um trabalho como este, jamais podemos afirmar que o resultado alcançado é um mérito individual.

Gostaria de agradecer ao meu pai, Francisco de Assis, e à minha mãe Maria Ivani, que me apoiaram e me incentivaram a todos os dias. Seja nos períodos difíceis do curso e nos momentos de felicidade que esboçava durante o curso.

Agradeço a minha amada, Larissa, por sempre me apoiar, ser o meu sustento muito das vezes, e se tornar meu incentivo e motivação do meu foco durante esta longa jornada.

Agradeço aos meus colegas, que percorreram a jornada junto a mim durante estes cincos anos juntos. Agradeço aos meus amigos do Centro Acadêmico que lutaram comigo durante toda essa jornada.

Aos meus professores, agradeço por serem meus colegas, e muitas vezes um amigo com quem contar. Ao Prof. Dr. Luís Fernando, meu mentor e amigo desde o início do curso, agradeço por muitos ensinamentos e orientações dadas a todos os momentos em conjunto. Sua orientação nunca será apenas durante a entrega de trabalho, e sim durante toda a vida.

Por fim, quero agradecer a todos que participaram desta jornada, e que um dia possam desfrutar deste trabalho, para ensinamentos póstumos.

RESUMO

A Inteligência Artificial é um campo da tecnologia moderna que se dedica majoritariamente ao desenvolvimento de sistemas que podem raciocinar, aprender e tomar decisões de forma autônoma, sendo assim o aprendizado de máquinas é uma área da IA que se concentra no desenvolvimento de algoritmos que podem aprender com dados sem serem capacitados a isso previamente. As bibliotecas de Aprendizado de Máquina para Python são ferramentas essenciais para o desenvolvimento de aplicações nessas áreas, pois fornecem uma ampla gama de funcionalidades e vantagens, permitindo assim que desenvolvedores tenham seu ponto de início de forma facilitada. Este trabalho apresenta uma análise de duas bibliotecas disponíveis para Python, destacando suas características, funcionalidades e vantagens. Esse estudo promove a explicitação de como funciona as bibliotecas, suas adequações para cada tipo de problema, e suas diferenciações entre si. A escolha das duas bibliotecas mais completas até então para o uso em *softwares* faz com que o trabalho tenha um grau de informação mais completo para o público que usa a linguagem *Python* como ferramenta de trabalho. Além disso é possível notar que cada biblioteca possui vantagens diferentes das outras, podendo fazer com que o programador possua a capacidade de mesclagem para extração completa de todos os benefícios. Pela relevância do tema, é possível notar que existe pouca documentação voltada especificamente a informação exclusiva de cada função dentro de cada biblioteca existente.

Palavras – chave: *Python*, Inteligência Artificial, Aprendizado de Máquina, *Software*, Biblioteca.

ABSTRACT

Artificial Intelligence is a field of modern technology that is mostly dedicated to the development of systems that can reason, learn and make decisions autonomously. Machine learning is an area of AI that focuses on the development of algorithms that can learn from data without being trained to do so beforehand. Machine Learning libraries for Python are essential tools for developing applications in these areas, as they provide a wide range of functionalities and advantages, thus allowing developers to get their start easily. This undergraduate work presents an analysis of two libraries available for Python, highlighting their characteristics, functionalities, and advantages. This study explains how the libraries work, their suitability for each type of problem, and their differences from one another. The choice of the two most complete libraries so far for use in software means that the work has a more complete level of information for the public that uses the Python language as a working tool. In addition, it is possible to see that each library has different advantages from the others, so that the programmer can mix them to fully extract all the benefits. Given the relevance of the topic, it is possible to note that there is little documentation specifically aimed at the unique information of each function within each existing library.

Keywords: Python, Artificial Intelligence, Machine Learning, Software, Library.

LISTA DE ABREVIATURAS

API	Application Programming Interface
CNN	Redes Neurais Convolucionais
DBSCAN	Density-Based Spatial Clustering Of Applications With Noise
FAIR	Facebook Ai Research
KNN	K-Nearest Neighbors
LDA	Linear Discriminant Analysis
NB	Naive Bayes
PLN	Processamento De Linguagem Natural
PLSR	Partial Least Squares Regression
RNA	Redes Neurais Artificiais
SVM	Support Vector Machine
SVR	Support Vector Regression
GPU	Graphics Processing Unit

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

O avanço do estudo em Aprendizado de Máquina (*Machine Learning,* ML) dentro da Inteligência Artificial tem sido de alto impacto na sociedade moderna devido seu desempenho na solução de uma ampla gama de problemas em diversos campos: tecnologia, saúde, ciências humanas, dentre outros. O ML é um particionamento da inteligência artificial (IA) onde o foco no desenvolvimento de algoritmos e técnicas permitem que os computadores possuam a capacidade de aprendizado e melhoria com base em dados e informações disponibilizadas a eles (Géron, 2019).

Com o avanço da capacidade computacional e desenvolvimento da tecnologia moderna, com o acesso a grande massa de dados de todas fez com que exponencialmente o progresso ML ocorresse. Algoritmos especializados e com alto grau de complexidade, impulsionados pela arquitetura de hardware especializada, permitiram com que melhorias de alto impacto no desempenho em tarefas como processamento de linguagem natural, tradução simultânea e reconhecimento de imagem ocorressem.

Softwares como o *Chat Generative Pre-trained Transformer*, popularmente conhecido como chat *GPT*, onde a interação entre humano-máquina acontece fornecendo soluções em texto para diferentes questionamentos e requisitos realizados.

O ML, com o avanço atual, permitiu a potencialização na automatização de tarefas repetitivas e exaustivas ao ser humano, aprimorando assim o tempo e recursos para atividades de caráter estratégico e criativo, onde atualmente a tecnologia possui uma defasagem na aplicação específica.

Tomando como exemplo, o processo do *software* citado anteriormente, permitiu que respostas fossem geradas e soluções a problemas comuns do cotidiano fossem realizadas em poucos segundos, reduzindo assim a carga de trabalho dos agentes humanos na hora da busca.

A criação de bibliotecas linguagens de programação voltadas a Inteligência Artificial (IA) pode ser reconhecida, também, como um dos fatores que impulsionaram

o uso e a implementação de algoritmos de ML. Essas bibliotecas disponibilizam uma variedade de funções e ferramentas e algoritmos pré-implementados, fazendo com que os desenvolvedores e cientistas de dados aproveitam as capacidades de ML, sem que uma criação seja gerada do zero repetidas vezes (Raschka.; Mirajalili, 2017).

Outro fator relevante está relacionado ao surgimento de *frameworks* de IA contribuindo para a disponibilização de um conjunto de ferramentas, estruturas e abstrações, facilitando o desenvolvimento e compreensão de aplicações de ML. Esses *frameworks* incluem dentro dela bibliotecas tais como explicitadas anteriormente, mas garantem também uma estrutura globalizada para criação de modelos de treinamento, avaliação, implantação e testes (Coelho *et al.*, 2023).

1.1 TRABALHOS CORRELATOS

O avanço da IA teve seu progresso alinhado juntamente com a necessidade da sociedade de facilitar suas atividades e reconhecer padrões nas informações expostas a ela. A literatura apresenta alguns trabalhos relacionados a estudos de bibliotecas ou *frameworks* ou algoritmos de ML, sendo alguns citados nos parágrafos seguintes.

Junto às criações de Processamento de Linguagem Natural (PLN), um campo que pode ser destacado refere-se aos algoritmos de redes neurais de aprendizado profundo, permitindo desenvolvimentos tais como o proposto por Chaves (2018), no qual o reconhecimento facial utiliza o *framework Tensorflow,* realizando uma comparação com demais *frameworks* com relação a velocidade de treinamento.

Bhatia (2017) propõe um estudo sobre velocidade de treinamento, com testes dos *frameworks TensorFlow* versão 1.3.0, *MXNet* versão 1.2.2, *Theano* versão 0.9.0, utilizando como *frontend* o *software Keras* - na versão 2.0.8.

A introdução ao uso de bibliotecas Python voltadas para criptografia de dados é interessante do ponto de vista de Rodrigues e Binda (2019), que abordam as formas de usar a biblioteca PyCripto e suas formas de criptografia e a simplicidade de usar um *framework* para auxílio ao desenvolvedor.

Carmo (2017) abordou duas bibliotecas SWING e JAVAFX para JAVA, destacando em seu trabalho que essas duas bibliotecas gráficas possuem um alto potencial na qualidade da interface do *Software* no quesito interface-usuário, relacionado a uma pesquisa com 51 usuários e profissionais da área de tecnologia sobre a opinião a estas bibliotecas.

1.2 OBJETIVOS

1.2.1 Objetivo geral

O presente trabalho em questão apresenta um estudo sobre as bibliotecas mais conhecidas de ML disponíveis para a linguagem de programação *Python*, focando em suas potencialidades, limitações, e como é o seu funcionamento dentro das soluções a serem futuramente implementadas.

1.2.3 Objetivos específicos

Os objetivos específicos deste trabalho concentram-se nos seguintes tópicos:

- O estudo sobre as principais bibliotecas para aplicação de ML disponíveis para a linguagem *Python*.
- Apresentação de alguns algoritmos disponíveis para as tarefas de classificação, agrupamento, regressão e regra de associação nas bibliotecas *Scikit-Learn* e *PyTorch*.
- Análise sobre as vantagens e limitações de cada biblioteca.

1.3 ORGANIZAÇÃO DO TRABALHO

Essa monografia está dividida em cinco capítulos. O primeiro Capítulo contém uma breve contextualização sobre o tema proposto nesta pesquisa e os respectivos objetivos. O segundo Capítulo apresenta a fundamentação teórica necessária para o entendimento do conteúdo abordado pela monografia. O terceiro Capítulo contém os materiais e métodos utilizados para o desenvolvimento da pesquisa. O quarto

Capítulo apresenta os testes e respectivos resultados. O quinto Capítulo finaliza com a conclusão do trabalho, comentando os resultados anteriormente obtidos.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são apresentados conceitos e características dos assuntos que permitirão o desenvolvimento deste trabalho. As bibliotecas que serão utilizadas para usarem de formas de medição comparativa, o *dataset* e os conhecimentos gerais sobre *Machine Learning* em *Python* acerca de *frameworks* e Inteligência Artificial, discutindo tanto a importância do tema, como as técnicas que são utilizadas para as medições utilizadas.

2.1 INTELIGÊNCIA ARTIFICIAL E APRENDIZADO DE MÁQUINA

É possível definir IA, como o estudo de agentes que recebem percepções do ambiente e executam ações. Cada fator é utilizado para a implementação de uma função que é capaz de mapear sequências de percepções em ações, e de tal forma diferentes maneiras de representar essas funções, tais como sistemas de produção, agentes reativos, planejadores condicionais em tempo real, redes neurais e sistemas de teoria de decisão (Russell; Norvig, 2021). Sua criação impactou fortemente em âmbito tecnológico, e sentiremos toda a consequência dela na próxima década, no âmbito de avanços da computação, e outras áreas como: medicina, economia e educação.

Machine Learning, ou aprendizado de máquina, é um ramo da inteligência artificial que se baseia na ideia de que sistemas de computador, onde são capazes de aprender com dados, identificar padrões e tomar decisões com pouca intervenção e ação humana. Em 2023, tornou-se essencial na tecnologia e em grande parte da sociedade moderna. Este tipo de conhecimento explora a construção de algoritmos que aprendem com seus erros e fazem previsões sobre dados a partir de diferentes abordagens (Bianchi, 2020).

2.1.1 Aprendizado supervisionado

No aprendizado supervisionado, algoritmos são utilizados para induzir modelos preditivos por meio da observação de um conjunto de objetos rotulados

(Luxburg; Schölkopf, 2011), conhecidos como base de treinamento ou *training set.* As nomeações contidas em tal conjunto correspondem a classes ou valores obtidos por alguma função desconhecida. Desse modo, um algoritmo de classificação buscará produzir um classificador capaz de generalizar as informações contidas no conjunto de treinamento, com a finalidade de classificar, posteriormente, objetos cujo rótulo seja desconhecido (Padilha; Carvalho, 2017).

De forma simplificada, o processo de aprendizado supervisionado pode ser comparado com o processo de aprendizagem de um aluno em uma sala de aula com um professor para auxílio. Ao resolver determinada tarefa, o professor avalia a resposta informando qual a resposta correta. Após a repetição contínua deste processo, o aluno saberá responder a novas questões corretamente, sem o auxílio do professor (Ferreira, 2016).

2.1.2 Aprendizado de máquina não-supervisionado

No aprendizado supervisionado já se conhece desde o começo a saída esperada e através dela se desenvolve o conhecimento até obter o resultado esperado. Em contrapartida, no aprendizado de máquinas não supervisionado o algoritmo atua sem um supervisor, avaliando as saídas. Nestes algoritmos não existe influência humana direta e o conhecimento se dá a partir do agrupamento dos dados baseados em suas similaridades (Nunes, 2018).

No caso dos algoritmos de aprendizado de máquina não-supervisionado, não é criado o vínculo entre um rótulo para os dados de saída. Com base em um número grande de dados, o algoritmo busca padrões e similaridades entre os dados, permitindo identificar grupos de itens similares ou similaridade de itens novos com grupos já definidos (Fontana, 2020).

2.2 ALGORITMOS DE APRENDIZADO DE MÁQUINA

Em tese relativa a Khaidem, Saha e Dey (2016), diferentes algoritmos têm sido utilizados para prever o sentido das ações, como SVM (*support vector machine*), redes neurais artificiais, regressão linear, KNN (*K-Nearest Neighbors*) e o

classificador Naive Bayes. Isto posto, a literatura revela que o SVM tem sido usado na maioria das vezes nesse tipo de pesquisa.

Neste particionamento do trabalho será abordado, também, sobre o J48 e o RF (*Random Forest)*, dois algoritmos de *ensemble learning* que permaneceram pouco abordados no problema de previsões e determinar o sentido ações, segundo Khaidem, Saha e Dey (2016).

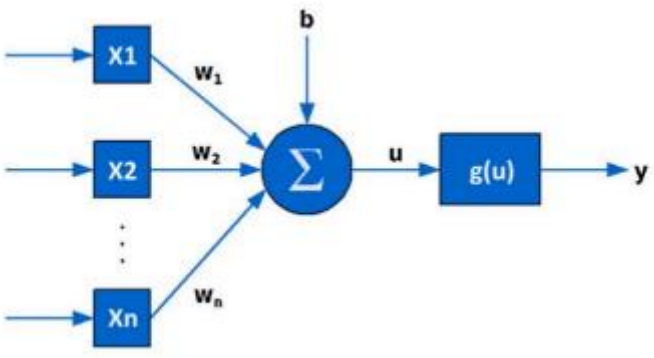
2.2.1 Redes neurais artificiais

Bishop (2006) define Rede Neural Artificial (RNA) no processo baseado nas tentativas de representar processamento matemático de informações em sistemas biológicos.

Uma das principais características das RNA é a habilidade de se desenvolver não só por meio de exemplos, mas também de aumentar sua capacidade com todas as informações aprendidas anteriormente, e assim sucessivamente.

Pode-se compreender essa formação de neurônios da rede, conforme a Figura 2.1, onde é figurado de forma baseada a um modelo matemático onde um número x de entradas ($x^1, x^2, x^3, x^{n-1} e\ x^n$) de forma que representaria os dendritos do cérebro humano, e y representando o axônio.

Figura 2.1: Perceptron



Fonte: Laboissiere, Fernandes e Lage (2015).

O comportamento do cérebro é realizado através de sinapses ($w^1, w^2, w^3, w^{n-1} e\ w^n$), podendo se comportar de forma que possua valores positivos

ou negativos. Já o viés do neurônio é representado por um valor limiar b. A função de ação de g é responsável pelo processo do que é recebido através das informações e possui a responsabilidade de fornecer a saída do neurônio (Laboissiere; Fernandes; Lage, 2015).

De acordo com Laboissiere, Fernandes e Lage (2015), o neurônio artificial assim como explicitado anteriormente, é realizada da seguinte maneira:

1. Os sinais são submetidos às entradas;
2. Cada sinal é multiplicado pelo seu respectivo peso sináptico;
3. É realizada uma soma entre os sinais de entrada ponderados e o neurônio;
4. O valor limite é calculado;
5. Esta informação é processada pela função de ativação do neurônio;
6. Um sinal de saída é produzido.

2.2.2 Naive Bayes

Naive Bayes (NB) é uma forma de classificação que utiliza como base o teorema de Bayes com uma suposição de independência entre os preditores. Um classificador NB assume que a existência de uma característica peculiar em uma classe não está em união com a existência de qualquer outro método. (Lira et al, 2023)

O teorema fornece uma forma de calcularmos a probabilidade futura P(A|B) a partir de P(A), P(B) e P(A|B). A equação 1 ilustra este cálculo (Lira et al., 2023), sendo P(A) é a probabilidade a priori, P(B|A) a probabilidade condicional e P(B) é a probabilidade total dos eventos ocorrerem.

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A) \cdot P(B|A)}{P(B)} \tag{1}$$

Dentre as vantagens do NB podem ser citadas a sua facilidade de entendimento e implementação, de forma simplificada, os programadores conseguem extrair de seu máximo com certa aptidão e clareza; sua diferença de possuir um processo de inferência mais veloz do que muitos outros classificadores citados aqui

neste trabalho, e sua capacidade de ser facilmente treinado com *datasets* pequenos; removendo a alta necessidade de possuir uma base de dados complexa e extensa para que NB seja aplicado (Kaviani; Dhotre, 2017).

2.2.3 J48

O algoritmo J48 possibilita o desenvolvimento de modelos de decisão em árvores. Este modelo de árvore de decisão é criado com base na análise dos dados de treino e pela modelagem utilizada para classificação de dados ainda não determinados. O algoritmo é capacitado para que seja possível a geração de árvores de decisão, a qual cada *node* da árvore avalia de forma parcial a presença ou a relevância de cada atributo de maneira individual (Frutuoso, 2014).

As árvores são criadas através da determinação do atributo mais aconselhável para a situação exposta e são criadas do topo dela para a sua base. Tavares, Bozza e Kono (2007) explicitou em seu trabalho que, o algoritmo é construído com uma árvore de decisão a partir do atributo mais significante a ela. por meio da abordagem *top-down* (topo-base).

Referente a isto, o atributo mais em destaque para o algoritmo é escolhido para que seja a raiz da árvore, igualando-o com todos os demais atributos do conjunto. Com isso, para continuidade da construção da árvore, é levado em conta que o segundo atributo como sendo o nó próximo da árvore, e assim até que se gere o nó folha, que determina o foco do atributo da instância gerada.

2.2.4 RANDOM FOREST

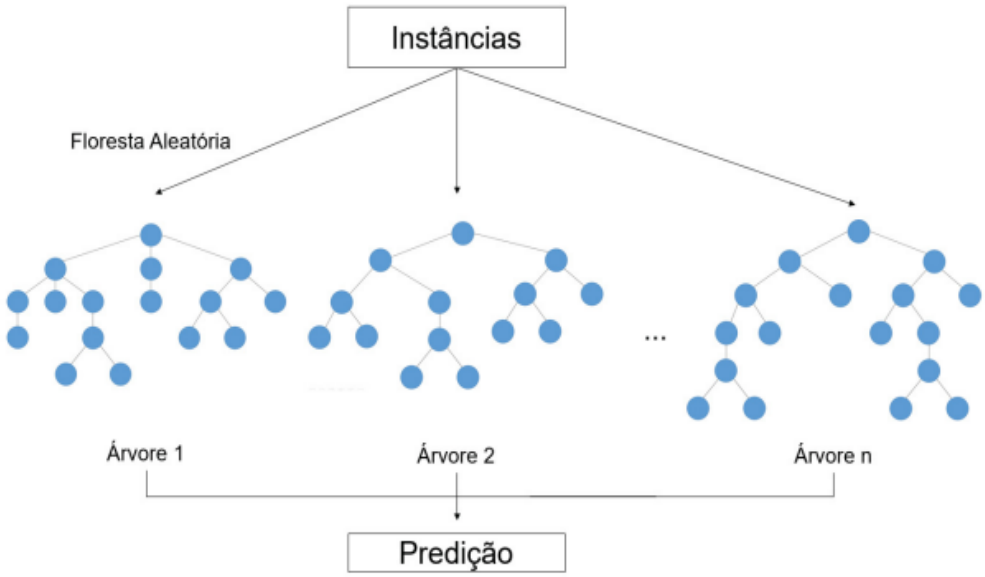
Random Forest (Floresta Aleatória) é um modo de aprendizado onde une a classificação e regressão de forma que é construído várias árvores de decisão através do treinamento e assim produzindo a classe, onde é o método de que as saídas geradas por árvores individuais. O algoritmo amplamente conhecido foi criado por Leo Breiman e Adele Cutler (Breiman, 2001).

O termo foi criado por conta de florestas de decisão aleatória (*Random Decision Forest*), gerado por Tin Kam Ho em 1995.

Cada árvore que tem seu uso na floresta aleatória é definida com árvore de regressão que possui a responsabilidade de aprimorar um número parametrizável de atributos, empregados de forma randômica em cada treinamento, para seu desenvolvimento (Breiman, 2001).

Na Figura 2.2 está demonstrando um modo de visualizarmos uma floresta aleatória, onde os caminhos de entrada são apresentados em todas as árvores espalhadas pela floresta. Os valores recebidos são então postos à frente da média, que é o valor final predestinado pela floresta. O propósito é o crescimento da variação do conjunto e diminuir a tendência dos estimadores, guiando assim um melhor modelo em relação ao uso de árvores isoladas. Assim, significando que durante as predições usando-se apenas uma árvore que é sensibilizada aos ruídos no conjunto de treinamento, a média indicada anteriormente não é sensível devida a não correlação entre elas (LOPES,2021).

FIGURA 2.2: Funcionamento de uma floresta aleatória.



Fonte: Lopes (2021)

Assim, além da obtenção das amostras de treinamento, o uso de algoritmo RF também é capacitado a receber amostras do conjunto de características, para a construção da árvore. Denominado como *bagging* de características (*feature bagging)*, seu conceito é a diminuição da dependência entre as árvores geradas e

para as árvores de regressão de características, (Hastie, 2003)) indica que 3 é o número de características em cada árvore.

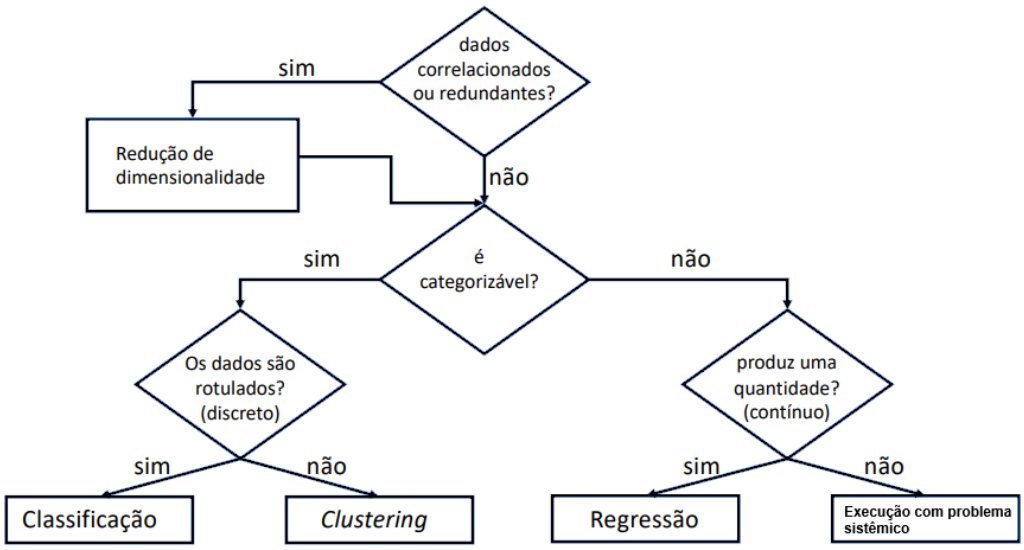
2.3 CLASSIFICAÇÃO DOS ALGORITMOS EM MACHINE LEARNING

O Machine Learning se baseia na ideia de que computadores tomem decisões com base em estatística junto de algoritmos. Esses algoritmos, produzidos até então, reconhecem padrões e fazem predições. Com o aumento de algoritmos, possuímos uma tipagem de problemas que podemos classificar atualmente em quatro subáreas principais: Classificação; Regressão; Agrupamento; Regra de Associação.

A principal segregação em relação ao paradigma de *machine learning*, válido para qualquer tipo de sistemas com disponibilidade da sua adaptação, é o conceito de aprendizagem não-supervisionada e aprendizagem supervisionada. (Vieira Junior, 2013)

Neste trabalho será abordado sobre as características e o que define cada um e as distinções entre si. Uma forma gráfica de abordar o uso e a separação é o fluxograma que será apresentado na Figura 2.3.

Figura 2.3:Fluxograma para discernimento das abordagens voltadas a machine learning



Fonte: Lima Neto (2022)

2.3.1 Classificação

Os algoritmos de classificação possuem sua base em prever a categoria de um dado apresentado, onde é procurada a estimativa de “classificador” que produza como resultado a classificação qualitativa de uma informação não observada com base em dados de entrada (abrangendo observações das classificações previamente definidas) (Silva, 2020).

De forma primária, o algoritmo é aprimorado com um conjunto de dados com classes conhecidas, sendo que estes dados podem estar separados em somente duas (classificação binária) ou em múltiplas classes (classificação multiclasse) (Fontana, 2020).

A categorização de um procedimento de discriminação de entidades concretas ou abstratas em grupos ou categorias, ou de maneira resumida, efeito ou método de distribuir por grupos. Se estas classes estiverem montadas e existirem uma informação sobre a possibilidade de um determinado objeto existir dentro da classe, é então classificado como informação inicial completa, se não, esta informação inicial estará ainda em produção. Com base na tipagem de informação inexistente o passo seguinte é o uso e a escolha do método. Métodos são geralmente classificados em paramétricos e não paramétricos (Vieira Junior, 2013).

Nos métodos paramétricos a distribuição da população contida em todo *dataset* tem uma forma padronizada e os encadeamentos, condicionados por esta tese, dizendo em si, a um ou vários parâmetros, como exemplos, Modelos Discriminantes e Regressão Linear. Por outro lado, nos métodos não paramétricos: a forma da entrega da população é desconhecida e os encadeamentos processam-se em uma formulação muito menos restrita e diversas vezes não se utilizam parâmetros.

2.3.2 Regressão

Regressão refere-se no algoritmo que se baseia em antecipar a saída de uma variável numérica (dependente) através da união de uma ou diversas variáveis independentes. Um cálculo de regressão é utilizado em estatística para encontrar qual

a relação é contida dentro dela, e se contém, no conjunto de dados (Machado, 2020).

O exemplo mais simplório deste tipo de modelagem é o comum ajuste de pontos a uma curva $y = f(x)$, onde é possível a associação a um único atributo x com uma saída unitariamente y. Entretanto, deste modo como ocorre na classificação, ocorre da necessidade da associação da saída com uma cadeia de atributos em um vetor x (Fontana, 2020).

2.3.2.1 Regressão linear

A missão principal dos algoritmos de regressão é a previsão do valor de uma ou mais variáveis contínuas y, comumente são denominadas de variáveis objetivas, tendo como ponto de partida um vetor x contendo n atributos reconhecidos. De tal modo, um conjunto de treinamento contendo m elementos $X = (x^1, x^2 \ldots . e\ x^m)^T$ unindo então os valores ligados das variáveis objetivo separadamente ao elemento, $y = (y^1, y^2 \ldots . e\ y^m)^T$, tendo como função a previsão do valor y^i correspondente ao novo vetor x^i(Fontana, 2020).

Da mesma forma que ocorre com a problematização de classificação, a regressão também envolve uma fonte de incerteza. Do ponto de vista probabilístico, um modelo de regressão é empregado para calcular a probabilidade condicional $p(y|x)$, ou seja, uma função que descreve como é realizada a distribuição de probabilidade da variável alvo, de forma que adquire o valor y quando fornecida uma entrada x (Fontana, 2020).

2.3.2.2 Regressão não-linear

Regressão não-linear, comumente, são conservados por alguma informação entre a relação de $Y\ e\ x$. Esta informação está unida a diferença de graus de conhecimento, como exemplo, uma análise de um diagrama de dispersão de y contra x; limitação de forma da função; e, também, a forma que é interpretada os seus parâmetros frente ao algoritmo. Independente do grau de conhecimento sobre algoritmos, a escolha de um modelo não linear dificilmente é empírica (Zeviani et al.,

2013).

Diferentemente da regressão linear, a não linear possui como foco principal, a presença de diferentes taxas de crescimento ao longo do seu curvamento, que lhe permitem uma caracterização de uma semi-parábola, podendo, de tal forma, um melhor representativo do comportamento das variáveis ao decorrer da curva.

Para otimizar algoritmicamente, considera-se onde o ajuste da curva que permite que ela minimize a soma dos quadrados dos resíduos, para tanto, é utilizado a abordagem dos mínimos quadrados ordinários. Isto é, o método que retorna os coeficientes que descrevem o comportamento da curva que inferioriza a distância entre os pontos e a reta, de desfrute destes é possível aplicá-los na função e descobrir as coordenadas da curvatura para cada ponto (Sousa, 2022).

2.3.3 Agrupamento

O Agrupamento de Dados é uma técnica algorítmica utilizada amplamente para mineração de dados de diversas formas, utilizando-se de metodologia numérica e a partir das informações das variáveis de cada dado disponibilizado, tendo como objetivo agrupar de forma sistêmica por aprendizado não supervisionado os n casos do *dataset* em m grupos, comumente disjuntos denominados *clusters*. Em diversas literaturas, o agrupamento de dados é denominado também como *Clusterização*, Taxonomia numérica e Análise de Agrupamentos (Cassiano, 2014).

Assim como determinado o conceito de classificação, o agrupamento é uma técnica mais “simplória” na qual quaisquer suposições são realizadas a respeito dos grupos de dados apresentados. De forma oposta, a *Clusterização* não conta com classes criadas de forma inicial e exemplificações de treinamento de classes de forma rotulada, assim sendo uma forma de aprendizado não supervisionado.

A vantagem principal do uso de técnicas de agrupamento é onde, ao juntar dados semelhantes, é possível a descrição de formas mais eficientes e eficazes as características ímpares de cada um dos grupos identificados. Assim fornecendo uma gama ampla de entendimento do *dataset* original, além de possibilitar a criação de esquemas de classificação para novos dados e encontrar junções interessantes entre

os atributos dos dados que não seriam descobertas sem o emprego de tais técnicas. De forma alternativa, agrupamento pode ser utilizado também como uma etapa de pré-processamento para os demais algoritmos, tais como caracterização e classificação, que se executam nos *clusters* identificados. (Cassiano,2014)

Uma definição formalizada da dificuldade de *Clusterização* é visualizada em Hruschka e Ebecken (2001). Onde considera-se que um conjunto de n objetos $X = \{X^1, X^2, \ldots, X^n\}$ onde cada $Xi\, \varepsilon\, IR^p$ é um vetor de p medidas reais que dimensionam as características do objeto, assim devendo serem agrupados em k clusters disjuntos $C = \{C^1, C^2, \ldots, C^K\}$, de forma que possuímos as seguintes condições respeitadas:

1. $C_1 \cup C_2 \cup \ldots \cup C_k = X;$
2. $C_i \neq \emptyset, \forall i, 1 \leq i \leq k;$
3. $C_i \cap C_j = \emptyset, \forall i \neq j, 1 \leq i \leq k, 1 \leq j \leq k.$

Importa-se que, tais condições, um objeto não pode integrar-se a mais de um *cluster* e que cada agrupamento tem que ter ao menos um objeto.

2.3.4 Regra de associação

A regra de associação possui como ideia inicial a busca de elementos que indicam a presença de outros elementos em uma única troca, ou seja, descobrir ligações ou modelos frequentes entre *datasets*. O termo “única troca” indica quais itens foram consultados em uma determinada operação de consulta aos dados. (Vasconcelos e Carvalho, 2014).

Comumente, regras de associação demonstram padrões existentes em transações armazenadas. Como exemplificação, a partir de um *dataset*, no qual é registrado os itens adquiridos por compradores de uma determinada loja, uma estratégia de mineração, com o uso de regras de associação, pode produzir a seguinte regra: {motor, óleo} ⟶ {rodas}, a qual aponta que o comprador que compra motor e óleo, com um grau de afirmação, compra também um jogo de rodas. Este grau de afirmação de regras de associação é demonstrado por dois índices: o fator

de suporte e o fator de confiança (Vasconcelos e Carvalho, 2014).

As bases de dados envolvidas nestes processos são extensas. Assim, é necessário que o uso de algoritmos rápidos e eficientes seja aplicado.

Um exemplo muito conhecido em Mineração de Dados foi desenvolvido pela Wal-Mart, A empresa varejista descobriu que o perfil do consumidor de cervejas era semelhante ao de fraldas. Assim sendo, homens casados, na faixa etária de 25 e 30 anos, onde ao comprar cervejas e/ou fraldas às sextas-feiras após expediente de trabalho em seu caminho para casa. Com base na comprovação destas hipóteses, o Hipermercado optou por uma melhoria das atividades junto às gôndolas nos pontos de vendas, aproximando as fraldas ao lado das cervejas (Silva, 2004).

As definições formais da regra de associação e base de dados de forma transacional são apresentadas a seguir.

Seja $I = \{I^1, I^2, \ldots, I^n\ \}$ um determinado conjunto de itens. Seja F um *dataset* de transações, onde que cada transação *T* é montada por um conjunto de itens onde T $I\ I$. Unitariamente cada transação possui um identificador denominado TID. Uma regra de associação é uma consequência da forma A P B, onde *A* e *B* possuem a possibilidade de serem conjuntos por um ou múltiplos itens, A ∩ I, B ∩ I, e A ∪ B = AB. A é denominado de antecedente da regra e *B é* denominado de consequente.

Dada uma regra A Þ B, a sua medida de suporte (*Sup*) representa a proporção de transações do banco de dados que contêm os itens de A e B, indicando a relevância desses itens. Por outro lado, a sua medida de confiança (*Conf*) representa a proporção de transações que possuem os itens de A possuem os itens de B, mostrando a validez da regra. O problema da descoberta de regras de associação, tal como originalmente definido em 1993, consiste em encontrar todas as regras de associação com suporte e confiança maiores ou iguais a um suporte mínimo (*SupMin*) e uma confiança mínima (*ConfMin*), respectivamente, como especificado pelo usuário (Devmedia, 2023).

2.3.4.1 Regra de associação transacionais

As regras de associação mineradas a partir de bases de dados de transações (como a ilustrada na Figura 2.4). Elas são conhecidas na literatura comumente como regras de associação transacionais ou regras de associação convencionais.

FIGURA 2.4: Base de dados de Transações

TID	Produtos Comprados
1	biscoito, cerveja, chá, salaminho
2	cerveja, couve, lingüiça, pão, queijo
3	café, brócolis, couve , pão
4	brócolis, café, cerveja, couve, pão, salaminho
5	brócolis, café, couve, pão, refrigerante
6	couve, lingüiça

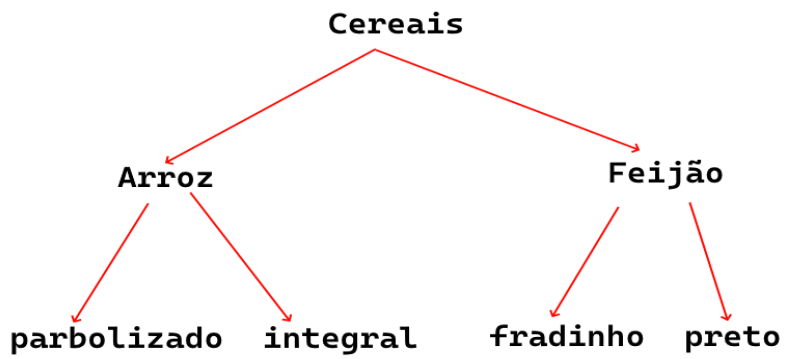
Fonte: Devmedia (2023)

Utilizando a base de dados da Figura L, tem-se que a seguinte regra pode ser garimpada: {*couve*} Þ {*brócolis*}. Esta regra é relevante, porque possui o auxílio de 50% (metade dos usuários (TID) que compraram ambos) e confiabilidade de 60% (indicando que 60% dos compradores que obtiveram couve também compraram brócolis).

2.3.4.2 Regra de associação multinível

Os itens de uma base de dados podem estar alocados em diferentes camadas, que os classificam hierarquicamente, conforme ilustrado na Figura 2.5.

Figura 2.5: Hierarquia de classificação



Fonte: *Devmedia (2023)*

Diversas ferramentas de Mineração de Dados possuem a capacidade de minerar regras de associação definidas não somente ao nível básico, mas também a partir de itens que representam classificações destes itens mais simplórios. Denominado como regra de associação multinível. Assim é possível minerar a regra genérica *{arroz} Þ {feijão}* tal como na imagem, sem a requisição da mineração de regras mais detalhadas como *{arroz integral} Þ {feijão preto}* e *{arroz parboilizado} Þ {feijão fradinho} (Devmedia, 2023).*

3 MATERIAIS E MÉTODOS

Este capítulo apresenta os materiais utilizados para a execução do processo de análise e o método utilizado.

3.1 TECNOLOGIA UTILIZADAS

A partir dos requisitos levantados e da análise de tecnologias foram definidas as tecnologias a serem utilizadas. A linguagem de programação *Python*, que surgiu no ano de 1991 desenvolvida pelo matemático Guido Von Rossum, é uma linguagem interpretada, de alto nível e fortemente tipada. Escolhida por sua forte integração com as demais ferramentas utilizadas (Rossum; Drake Junior, 1995).

Ela se destaca pela sua sintaxe clara e eficiente de conceitos bem definidos. Linguagem é capacitada a oferecer estruturas de dados avançadas, como listas, dicionários e manipulação de data/hora, além de uma vasta gama de módulos prontos para uso. Sua característica de tipagem dinâmica, combinada com a natureza interativa e interpretada, faz dela uma opção versátil, sendo o oposto de linguagens como JAVA, que são de tipagem padronizada, que dificulta o aprendizado e a execução de trabalhos que são de baixo nível.

Além disso, Python é orientada a objetos e se presta tanto para criação de scripts quanto para desenvolvimento ágil de aplicações em diversas áreas. A linguagem também é compatível com múltiplas bibliotecas, inclusive para sistemas de janela. Isso possibilita a adoção tanto do paradigma de programação orientada a objetos quanto do procedural (Borges, 2014).

3.1.1 *Frameworks*

O conceito de *frameworks* na literatura apresenta diferentes perspectivas e abordagens. Nesta tese, serão abordados os pontos de vista de alguns autores.

Segundo Fayad, Schmidt e Johnson (1999), um *Framework* é um conjunto de classes que colaboram para realizar uma responsabilidade específica dentro de um domínio de um subsistema da aplicação.

Os frameworks permitem que funcionalidades já implementadas possam ser reutilizadas pelos desenvolvedores na construção de suas aplicações, reduzindo o esforço de trabalho e o tempo gastos na criação dele.

Um fator a ser citado, é a existência do *NumPy,* uma biblioteca em python, que adiciona suporte para matrizes e vetores multidimensionais extensos, junto de uma coletânea de funções matemáticas de alto nível pré-moldadas para que sejam operados esses vetores. *SciPy* é necessitado para constituição de algoritmos onde necessitam a resolução de problemas científicos e matemáticos. *SciPy* é elaborado com base no pacote NumPy e possibilita a manipulação e desenvolvimento de visualizações dos dados com uma alta disponibilidade de comandos de alto nível (Scalco, 2021).

3.1.1.1 Scikit-Learn

O *Scikit-learn* é uma das bibliotecas mais populares em Python para aprendizado de máquina (machine learning) e análise de dados. Ele fornece uma ampla gama de algoritmos de aprendizado de máquina supervisionados e não supervisionados, bem como utilitários para pré-processamento de dados, validação cruzada, seleção de modelos e avaliação de desempenho (Scalco, 2021).

Alguns dos algoritmos de aprendizado de máquina que o *Scikit-learn* oferece incluem regressão linear e logística, árvores de decisão, florestas aleatórias, SVM, k-means e muitos outros. Além disso, o *Scikit-learn* tem uma interface de programação de aplicativos (API) consistente e fácil de usar, o que o torna uma escolha popular para muitos cientistas de dados e desenvolvedores de aprendizado de máquina (Ribeiro,2021).

O *Scikit-learn* também fornece ferramentas para pré-processamento de dados, incluindo codificação one-hot, normalização, escalonamento e redução de dimensionalidade. Oferecendo também um suporte para validação cruzada, seleção

de modelos e avaliação de desempenho, que são etapas importantes no desenvolvimento de modelos de aprendizado de máquina confiáveis e precisos.

3.1.1.2 TensorFlow

O TensorFlow é uma biblioteca de software livre e de código aberto que oferece uma ampla variedade de ferramentas e recursos para criar, treinar e implantar modelos de aprendizado de máquina e IA. Ele permite que os usuários criem e treinem redes neurais profundas e outros modelos de aprendizado de máquina usando uma variedade de algoritmos de otimização e técnicas de regularização (Géron, 2019).

Além disso, o TensorFlow suporta implantação de modelos em várias plataformas, como desktops, dispositivos móveis, nuvem e até mesmo em hardware dedicado, como unidades de processamento de tensor. Ele também possui uma interface de programação de aplicativos amigável para o desenvolvedor.

Outro recurso útil do TensorFlow é a capacidade de visualizar o processo de treinamento de modelos e resultados por meio de ferramentas de visualização, como TensorBoard, que permite visualizar graficamente as métricas de desempenho e ajustar os hiperparâmetros para melhorar a precisão dos modelos.

3.1.1.3 PyTorch

O PyTorch é uma biblioteca popular de aprendizado de máquina e inteligência artificial (AI) desenvolvida pela equipe do Facebook AI Research (FAIR). É uma biblioteca de software livre e de código aberto que fornece ferramentas para criação de redes neurais profundas e outras arquiteturas de aprendizado de máquina em Python (Stevens et al, 2020).

O PyTorch é conhecido por ser fácil de usar e permite que os usuários criem e treinem modelos de aprendizado de máquina de forma rápida e eficiente. Ele usa uma estrutura de gráfico de computação dinâmica, o que significa que os usuários podem alterar a estrutura do modelo durante o tempo de execução. Isso permite que os usuários experimentem diferentes arquiteturas de modelo e façam alterações rapidamente, sem a necessidade de reescrever todo o código.

Outro recurso útil do PyTorch é o PyTorch Lightning, que é uma estrutura leve para organizar código de aprendizado de máquina em módulos claros e reutilizáveis. Ele automatiza as tarefas comuns de treinamento de modelos, como inicialização, checkpointing e ajuste de hiperparâmetros, permitindo que os usuários se concentrem na criação de modelos de alta qualidade.

O PyTorch também oferece uma ampla gama de ferramentas de visualização e depuração, incluindo o TorchVision, que é uma biblioteca para processamento de imagens, e o TensorBoard, que permite visualizar graficamente as métricas de desempenho do modelo.

3.1.1.4 Theano

A biblioteca Theano é, especialmente, projetada para criar e otimizar redes neurais profundas e outros modelos de aprendizado de máquina de forma eficiente. A biblioteca é implementada em *Python* e pode ser executada em *CPU* ou *GPU*. Ela também fornece uma série de ferramentas para gerenciamento de memória, otimização de código e paralelização para garantir que os modelos sejam executados de forma rápida e eficiente (Bourez, 2017).

O Theano é conhecido por sua capacidade de representar modelos matemáticos complexos de forma eficiente e rápida, graças à sua capacidade de aproveitar a arquitetura da GPU. Além disso, a biblioteca permite que os usuários definam funções simbólicas e símbolos de variáveis em *Python*, o que permite a criação e manipulação de modelos complexos de forma fácil e intuitiva.

Theano também é compatível com várias outras bibliotecas de aprendizado de máquina, incluindo scikit-learn, keras e tensorflow. Ele fornece uma ampla gama de ferramentas e recursos para modelagem de dados, visualização de resultados e solução de problemas.

No entanto, deve-se notar que, embora o Theano ainda seja uma biblioteca de computação numérica e simbólica poderosa, ela não é mais ativamente mantida desde 2017, e muitos usuários estão mudando para outras bibliotecas de aprendizado de máquina e deep learning mais modernas, como *PyTorch* e *TensorFlow*.

3.1.1.5 PyCaret

PyCaret é uma biblioteca de aprendizado de máquina de código aberto para Python que permite aos usuários criar modelos de aprendizado de máquina com poucas linhas de código. É uma ferramenta útil para cientistas de dados que desejam acelerar o processo de desenvolvimento de modelos e criar protótipos rapidamente (Moez, 2020).

O PyCaret oferece uma ampla gama de algoritmos de aprendizado de máquina, como regressão linear, regressão logística, árvores de decisão, florestas aleatórias, entre outros. Ele também possui recursos que ajudam na pré-processamento de dados, como tratamento de valores ausentes e codificação de variáveis categóricas. Além disso, ele fornece recursos de visualização de dados e avaliação de modelos, como validação cruzada, matriz de confusão e curvas de aprendizado.

Outra vantagem do PyCaret é a possibilidade de usar várias técnicas de aprendizado de máquina em uma única linha de código. Por exemplo, é possível criar um pipeline de processamento de dados e treinamento de modelos com apenas algumas linhas de código.

Em resumo, o PyCaret é uma biblioteca de aprendizado de máquina útil e eficiente para cientistas de dados e desenvolvedores que desejam acelerar o processo de desenvolvimento de modelos de aprendizado de máquina.

4 TESTES E RESULTADOS

Nesta subseção são apresentados os tipos de algoritmos aplicados em bibliotecas na linguagem Python, juntamente a suas respectivas disponibilidades e uma consideração final para cada uma das bibliotecas. Cada subseção abordará uma biblioteca diferente estuda e, no fim, uma subseção para considerações pertinentes sobre as mesmas e uma tabela de explanação referente a possibilidades de cada abordagem.

4.1 SCIKIT-LEARN

Observa-se que a biblioteca *Scikit-learn* classifica-se com as melhores notas para facilitar o desenvolvimento em *Python*, em foco porque os algoritmos funcionam como documentado em seu compêndio, as APIs são moldadas de forma a serem bem projetadas e de fácil uso, e há poucas “incompatibilidades de impedâncias” entre estruturas de dados.

Em viés, como a biblioteca não possui vertente ao *Deep Learning* ou aprendizado por reforço, deixa de fora os atuais problemas difíceis, mas importantes, como a classificação precisa da imagem e a análise e tradução confiáveis em tempo real.

4.1.1 Classificação

Em análise a documentação da própria biblioteca, podemos notar que para classificação, a biblioteca possui diversos algoritmos pré-moldados, assim facilitando que seja aplicado durante a programação em Python. De forma breve é possível explicitar que alguns desses algoritmos de classificação mais comuns no *Scikit-learn,* juntamente a breve descrição da sua aplicabilidade.

4.1.1.1 Regressão logística

A regressão logística é um algoritmo de classificação que habitualmente é utilizado principalmente para problemas de classificação binária, embora também possa a vir estendido para classificação multiclasse. Esse algoritmo possui a possibilidade de modelar a relação entre as variáveis de entrada e a probabilidade de pertencer a uma classe específica. O Quadro 4.1, a seguir, apresenta as principais funções relacionadas a este classificador.

Quadro 4.1: Funções do classificador Regressão Logística

Nome do método	Descrição
LogisticRegression	Método para construção de um classificador de regressão logística, que é um modelo de aprendizado de máquina supervisionado que usa uma função logística para converter as probabilidades em previsões categóricas.

4.1.1.2 k-vizinhos mais próximos (KNN)

O algoritmo KNN possibilita a classificação dos pontos de dados em base na classe predominante entre seus "k" vizinhos mais próximos, tal como o próprio nome é proposto. É um algoritmo simples e não paramétrico, que pode ser usado tanto para classificação quanto para regressão. O Quadro 4.2, a seguir, apresenta as principais funções relacionadas a este classificador.

Quadro 4.2: Funções do classificador K-Vizinhos Mais Próximos

Nome do método	Descrição
KNeighborsClassifier	Método para construção de um classificador KNN, que é um modelo de aprendizado de máquina supervisionado que classifica um novo exemplo com base nos k exemplos mais próximos dele no espaço de características

4.1.1.3 Máquinas de vetores de suporte (S*upport Vector Machines, SVM*)

As *SVMs* são eficazes para problemas de classificação binária. Elas retroagem para encontrar um hiperplano de separação que maximize a margem entre

as classes. As SVMs podem ser estendidas para classificação multiclasse por meio de técnicas como *One-vs-All (OvA)* ou *One-vs-One (OvO).* O Quadro 4.3, a seguir, apresenta as principais funções relacionadas a este classificador.

Quadro 4.3: Funções do classificador SVM

Nome do método	Descrição
SVC	Método para construção de um classificador SVM, que é um modelo de aprendizado de máquina supervisionado que usa uma função de separação não linear para separar dados de duas ou mais classes.
LinearSVC	Método para construção de um classificador SVM linear, que é um modelo de aprendizado de máquina supervisionado que usa uma função de separação linear para separar dados de duas classes.
NuSVC	Método para construção de um classificador SVM com kernel linear, que é um tipo de SVM que usa um kernel linear e um conjunto de suporte reduzido.

4.1.1.4 Árvores de decisão

As árvores de decisão são estruturas de árvores que dividem os dados em nós (folhas) com base em critérios de divisão. Elas são usadas para decisões de classificação com base em atributos. Pode haver sobreajuste se não forem controlados adequadamente. O Quadro 4.4, a seguir, apresenta as principais funções relacionadas a este classificador.

Quadro 4.4: Funções do classificador Árvores de Decisão

Nome do método	Descrição
DecisionTreeClassifier	Método para construção de um classificador de árvore de decisão, que é um modelo de aprendizado de máquina não paramétrico que usa uma árvore de decisão para classificar dados.
ExtraTreesClassifier	Método para construção de um classificador de árvores extrator aleatória, que é um tipo de árvore de decisão que usa bootstrapping aleatório e seleção aleatória de atributos para construir cada árvore.

4.1.1.5 Floresta aleatória

Uma floresta aleatória é uma coleção de árvores de decisão, geralmente usada para melhorar a generalização e reduzir o sobreajuste em relação a uma única árvore. Cada árvore é treinada em uma amostra aleatória dos dados e as previsões

são agregadas. O Quadro 4.5, a seguir, apresenta as principais funções relacionadas a este classificador.

Quadro 4.5: Funções do classificador Floresta Aleatória

Nome do método	Descrição
RandomForestClassifier	Método para construção de um classificador de floresta aleatória, que é um modelo de aprendizado de máquina ensemble que consiste em um conjunto de árvores de decisão treinadas com dados aleatórios.
ExtraTreesClassifier	Método para construção de um classificador de árvores extrator aleatória, que é um modelo de aprendizado de máquina ensemble que consiste em um conjunto de árvores de decisão treinadas com dados aleatórios e com a técnica de extração de atributos.

4.1.1.6 Gradient Boosting

Gradient Boosting é uma técnica que cria um modelo forte combinando várias árvores de decisão fracas (geralmente chamadas de estimadores fracos). O algoritmo popular baseado em *Gradient Boosting* é o *Gradient Boosting Classifier*, mas também existem variações como *AdaBoost* e *XGBoost*. O Quadro 4.6, a seguir, apresenta as principais funções relacionadas a este classificador.

Quadro 4.6: Funções do classificador Gradient Boosting

Nome do método	Descrição
GradientBoostingClassifier	Método para construção de um classificador de gradiente boosting, que é um modelo de aprendizado de máquina ensemble que consiste em um conjunto de árvores de decisão treinadas sequencialmente, onde cada árvore é treinada para corrigir os erros da árvore anterior.
HistGradientBoostingClassifier	Método para construção de um classificador de gradiente boosting baseado em histograma, que é um tipo de gradiente boosting que usa histogramas para representar os dados.

4.1.1.7 Redes neurais artificiais

As RNA são modelos de aprendizado profundo que consistem em várias camadas de neurônios artificiais. Elas são usadas para tarefas de classificação complexas, mas geralmente requerem grandes quantidades de dados e poder computacional. O Quadro 4.7, a seguir, apresenta as principais funções relacionadas a este classificador.

Quadro 4.7: Funções do classificador Redes Neurais Artificiais

Nome do método	Descrição
MLPClassifier	Método para construção de um classificador de rede neural multicamadas, que é um modelo de aprendizado de máquina supervisionado que usa uma rede neural multicamadas para classificar dados.
MLPClassifierCV	Método para construção de um classificador de rede neural multicamadas com validação cruzada, que é um modelo de aprendizado de máquina supervisionado que usa uma rede neural multicamadas para classificar dados usando validação cruzada.

4.1.1.8 Naive Bayes

Os classificadores Naive Bayes são baseados no teorema de Bayes e são eficazes para classificação de texto e problemas de classificação multiclasse. Eles assumem independência condicional entre os atributos, daí o termo "naive" (ingênuo). O Quadro 4.8, a seguir, apresenta as principais funções relacionadas a este classificador.

Quadro 4.8: Funções do classificador Naive Bayes

Nome do método	Descrição
BernoulliNB	Método para construção de um classificador Naive Bayes Bernoulli, que é um modelo de aprendizado de máquina supervisionado que usa uma distribuição Bernoulli para calcular a probabilidade de cada classe.
MultinomialNB	Método para construção de um classificador Naive Bayes Multinomial, que é um modelo de aprendizado de máquina supervisionado que usa uma distribuição multinomial para calcular a probabilidade de cada classe.

4.1.1.9 Classificador quadrático gaussiano

Uma variação do Naive Bayes que assume que os atributos numéricos seguem uma distribuição gaussiana. O Quadro 4.9, a seguir, apresenta as principais funções relacionadas a este classificador.

Quadro 4.9: Funções do classificador Quadrático Gaussiano

Nome do método	Descrição
GaussianNB	Método para construção de um classificador Naive Bayes Gaussiano, que é um modelo de aprendizado de máquina supervisionado que usa uma distribuição normal para calcular a probabilidade de cada classe.

4.1.1.10 Classificador quadrático linear (*Linear Discriminant Analysis, LDA*)

LDA é um método estatístico que encontra uma combinação linear dos atributos que maximizam a separação entre as classes. É eficaz para problemas de classificação multiclasse. O Quadro 4.10, a seguir, apresenta as principais funções relacionadas a este classificador.

Quadro 4.10: Funções do classificador Quadrático Linear

Nome do método	Descrição
LinearDiscriminantAnalysis	Método para construção de um classificador LDA, que é um modelo de aprendizado de máquina supervisionado que usa uma função discriminante linear para separar dados de duas ou mais classes.

4.1.2 Regressão

A regressão assim como cada um de suas funções tem suas vantagens e suas desvantagens e possui uma adequação ideal para diferentes tipos de problemas de regressão. A escolha do algoritmo é variabilidade conforme seu conjunto de dados apresentado e dos objetivos pontuais do projeto a ser desenvolvido. Alguns desses algoritmos de regressão são apresentados nas subseções seguintes.

4.1.2.1 Regressão linear

A regressão linear é um dos algoritmos de regressão mais simples e amplamente utilizados. Ele modela a relação entre uma variável de saída (dependente) e uma ou mais variáveis de entrada (independentes) por meio de uma equação linear. A regressão linear simples lida com uma variável de entrada,

enquanto a regressão linear múltipla lida com múltiplas variáveis de entrada. O Quadro 4.11, a seguir, apresenta as principais funções relacionadas a este regressor.

Quadro 4.11: Funções do regressor linear

Nome do método	Descrição
LinearRegression	Método para construção de um regressor de regressão linear, que é um modelo de aprendizado de máquina supervisionado que usa uma função linear para estimar um valor contínuo.

4.1.2.2 Regressão Ridge

A regressão Ridge é uma extensão da regressão linear que adiciona uma penalização L2 à função de custo para evitar o sobreajuste (ou "overfitting"). Isso ajuda a regularizar o modelo, tornando-o menos sensível a valores atípicos e multicolinearidade. O Quadro 4.12, a seguir, apresenta as principais funções relacionadas a este regressor.

Quadro 4.12: Funções de regressão ridge

Nome do método	Descrição
Ridge	Método para construção de um regressor de regressão linear com regularização L2, que é um modelo de aprendizado de máquina supervisionado que usa uma função linear para estimar um valor contínuo, penalizando os coeficientes do modelo para evitar o overfitting.

4.1.2.3 Regressão Lasso

A regressão Lasso é semelhante à Ridge, mas usa uma penalização L1 em vez de L2. Ela é útil para seleção de recursos, pois tende a produzir modelos esparsos, zerando os coeficientes de recursos menos importantes. O Quadro 4.13, a seguir, apresenta as principais funções relacionadas a este regressor.

Quadro 4.13: Funções de regressão lasso

Nome do método	Descrição
Lasso	Método para construção de um regressor de regressão linear com regularização L1, que é um modelo de aprendizado de máquina supervisionado que usa uma função linear para estimar um valor contínuo, penalizando os coeficientes do modelo para evitar o overfitting e a seleção de características.

4.1.2.4 Regressão Elastic Net

A regressão Elastic Net combina as penalizações L1 e L2 da Ridge e da Lasso. Isso oferece um equilíbrio entre as duas técnicas e é útil quando há multicolinearidade e muitos recursos. O Quadro 4.14, a seguir, apresenta as principais funções relacionadas a este regressor.

Quadro 4.14: Funções de regressão lasso

Nome do método	Descrição
ElasticNet	Método para construção de um regressor de regressão linear com regularização L1 e L2, que é um modelo de aprendizado de máquina supervisionado que usa uma função linear para estimar um valor contínuo, penalizando os coeficientes do modelo para evitar o overfitting e a seleção de características.

4.1.2.5 Regressão de mínimos quadrados parciais

A PLSR é usada quando há multicolinearidade entre os recursos. Ela cria recursos (componentes) que são combinações lineares dos recursos originais e, em seguida, realiza a regressão linear nos componentes. O quadro 4.15, a seguir, apresenta as principais funções relacionadas a este regressor.

Quadro 4.15: Funções de Regressão de Mínimos Quadrados Parciais

Nome do método	Descrição
PLSR	Método para construção de um regressor de regressão de componentes principais, que é um modelo de aprendizado de máquina supervisionado que usa uma combinação linear de componentes principais para estimar um valor contínuo.
PLSRCV	Método para construção de um regressor de regressão de componentes principais para problemas de regressão com validação cruzada, que é um modelo de aprendizado de máquina supervisionado que usa uma combinação linear de componentes principais para estimar um valor contínuo, usando validação cruzada para escolher o melhor valor do número de componentes principais.

4.1.2.6 Regressão de árvore de decisão

Assim como as árvores de decisão são usadas para tarefas de classificação, elas podem ser usadas para tarefas de regressão. O algoritmo cria uma árvore de

decisão para prever valores contínuos em vez de classes. O Quadro 4.16, a seguir, apresenta as principais funções relacionadas a este regressor.

Quadro 4.16: Funções de Regressão de Árvore de Decisão

Nome do método	Descrição
DecisionTreeRegressor	Método para construção de um regressor de árvore de decisão, que é um modelo de aprendizado de máquina supervisionado que usa um conjunto de regras simples para estimar um valor contínuo.
DecisionTreeRegressorCV	Método para construção de uma árvore de decisão para problemas de regressão com validação cruzada, que é um modelo de aprendizado de máquina supervisionado que usa um conjunto de regras simples para estimar um valor contínuo, usando validação cruzada para escolher o melhor valor do parâmetro de profundidade máxima.

4.1.2.7 Regressão de floresta aleatória

A regressão da Floresta Aleatória é uma extensão da regressão de árvore de decisão que utiliza uma coleção de árvores de decisão para melhorar a precisão e reduzir o overfitting. O Quadro 4.17, a seguir, apresenta as principais funções relacionadas a este regressor.

Quadro 4.17: Funções de Regressão de Floresta Aleatória

Nome do método	Descrição
RandomForestRegressor	Método para construção de um regressor de floresta aleatória, que é um modelo de aprendizado de máquina supervisionado que usa um conjunto de árvores de decisão para estimar um valor contínuo.
RandomForestRegressorCV	Método para construção de um regressor de floresta aleatória para problemas de regressão com validação cruzada, que é um modelo de aprendizado de máquina supervisionado que usa um conjunto de árvores de decisão para estimar um valor contínuo, usando validação cruzada para escolher o melhor valor do parâmetro de número de árvores de decisão.

4.1.2.8 Regressão de Gradiente

A regressão de Gradiente é uma técnica que cria um modelo forte combinando várias árvores de decisão fracas (geralmente chamadas de estimadores fracos). O algoritmo popular baseado em Gradiente é o Gradient Boosting, mas

também existem variações como AdaBoost e XGBoost. O Quadro 4.18, a seguir, apresenta as principais funções relacionadas a este regressor.

Quadro 4.18: Funções de Regressão de Gradiente

Nome do método	Descrição
GradientBoostingRegressor	Método para construção de um regressor de boosting de árvores de decisão, que é um modelo de aprendizado de máquina supervisionado que usa um conjunto de árvores de decisão para estimar um valor contínuo. O algoritmo de boosting de árvores de decisão funciona construindo um conjunto de árvores de decisão, cada uma das quais é treinada para corrigir os erros da árvore anterior.
GradientBoostingRegressorCV	Método para construção de um regressor de boosting de árvores de decisão para problemas de regressão com validação cruzada, que é um modelo de aprendizado de máquina supervisionado que usa um conjunto de árvores de decisão para estimar um valor contínuo, usando validação cruzada para escolher o melhor valor dos parâmetros do algoritmo de boosting.

4.1.2.9 Regressão de máquinas de vetores de suporte (*Support Vector Regression*, SVR)

A SVR é uma extensão da SVM para problemas de regressão. Ela tenta encontrar uma função de regressão que minimize a diferença entre as previsões e os valores reais, enquanto ainda mantém uma margem de erro. O Quadro 4.19, a seguir, apresenta as principais funções relacionadas a este regressor.

Quadro 4.19: Funções de Regressão de Máquinas de Vetores de Suporte

Nome do método	Descrição
SVR	Método para construção de um suporte vetorial, que é um modelo de aprendizado de máquina supervisionado que usa suporte vetorial para estimar um valor contínuo.
SVRCV	Método para construção de um suporte vetorial para problemas de regressão com validação cruzada, que é um modelo de aprendizado de máquina supervisionado que usa suporte vetorial para estimar um valor contínuo, usando validação cruzada para escolher o melhor valor dos parâmetros do kernel.

4.1.3 Agrupamento

Scikit-learn traz para agrupamento diversas técnicas de aprendizado não-supervisionado envolvendo a organização de dados em grupos em base às características semelhantes entre eles. As subseções seguintes apresentam alguns dos algoritmos presentes na biblioteca.

4.1.3.1 K-Means

O K-Means é um algoritmo de agrupamento de particionamento. Ele divide os dados em k clusters, onde k é um número especificado pelo usuário. O algoritmo atribui pontos de dados aos clusters com base na proximidade dos centróides dos clusters. O Quadro 4.20, a seguir, apresenta as principais funções relacionadas a este agrupador.

Quadro 4.20: Funções de Agrupamento K-Means

Nome do método	Descrição
KMeans	Método para construção de um modelo de agrupamento *k-means*, que é um algoritmo de aprendizado de máquina não supervisionado que divide os dados em k grupos.
MiniBatchKMeans	Método para construção de um modelo de agrupamento k-means para problemas de agrupamento com *minibatching*, que é um algoritmo de aprendizado de máquina não supervisionado que divide os dados em k grupos usando *minibatching*.

4.1.3.2 Agrupamento hierárquico

O agrupamento hierárquico cria uma árvore de clusters, onde cada nó na árvore representa um cluster. Pode ser aglomerativo (começa com pontos de dados individuais como clusters e mescla clusters adjacentes) ou divisivo (começa com todos os pontos de dados em um único cluster e divide-os). O Quadro 4.21, a seguir, apresenta as principais funções relacionadas a este agrupador.

Quadro 4.21: *Funções de Agrupamento Hierárquico*

Nome do método	Descrição
AgglomerativeClustering	Método para construção de um modelo de agrupamento hierárquico aglomerativo, que é um método de agrupamento que começa com cada ponto de dados em seu próprio cluster e então combina os clusters com base em sua proximidade.

4.1.3.3 Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

O DBSCAN é um algoritmo de agrupamento baseado em densidade. Ele encontra clusters com base na densidade de pontos de dados. Pontos densos são considerados parte dele, enquanto pontos isolados são considerados ruídos. O quadro 4.22, a seguir, apresenta as principais funções relacionadas a este agrupador.

Quadro 4.22: Funções de Agrupamento DBSCAN

Nome do método	Descrição
DBSCAN	Método para construção de um modelo de agrupamento hierárquico baseado em densidade, que é um método de agrupamento que identifica clusters baseados em densidade.

4.1.3.4 Mean shift

O *Mean Shift* é um algoritmo de agrupamento que busca os modos (picos) na distribuição de dados. Ele move iterativamente os centróides dos clusters na direção do ponto de maior densidade até convergir para os modos de dados. O Quadro 4.23, a seguir, apresenta as principais funções relacionadas a este agrupador.

Quadro 4.23: Funções de Agrupamento Mean Shift

Nome do método	Descrição
MeanShift	Método para construção de um agrupador de deslocamento médio, que é um algoritmo de agrupamento não supervisionado que usa um deslocamento médio para estimar o centro de cada grupo.
MeanShiftCV	Método para construção de um agrupador de deslocamento médio para problemas de agrupamento com validação cruzada, que é um algoritmo de agrupamento não supervisionado que usa um deslocamento médio para estimar o centro de cada grupo, usando validação cruzada para escolher o melhor valor do parâmetro de bandwidth.

4.1.3.5 Agrupamento Espectral

O agrupamento espectral utiliza a matriz de similaridade entre pontos de dados para criar clusters. Ele mapeia os dados para um espaço de recursos de alta dimensão, onde os clusters são mais facilmente separáveis, e, em seguida, aplica um algoritmo de agrupamento padrão. O Quadro 4.24, a seguir, apresenta as principais funções relacionadas a este agrupador.

Quadro 4.24: Funções de Agrupamento Espectral

Nome do método	Descrição
SpectralClustering	Método para construção de um modelo de agrupamento hierárquico baseado em espectro, que é um método de agrupamento que usa a decomposição espectral para criar uma representação de dados que é então usada para agrupar pontos de dados.

4.1.3.6 Balanced Iterative Reducing and Clustering Using Hierarchies (BIRCH)

O algoritmo Birch é um algoritmo de agrupamento hierárquico que é eficiente em termos de memória. Ele usa uma estrutura de árvore balanceada para representar os clusters e é adequado para grandes conjuntos de dados. O quadro 4.25, a seguir, apresenta as principais funções relacionadas a este agrupador.

Quadro 4.25: Funções de Agrupamento Birch

Nome do método	Descrição
Birch	Método para construção de um modelo de agrupamento hierárquico baseado em buquê, que é um método de agrupamento que usa um esquema de construção de árvores para criar uma estrutura de dados hierárquica.

4.1.3.7 Affinity Propagation

O Affinity Propagation é um algoritmo de agrupamento que encontra automaticamente o número de clusters com base na similaridade entre os pontos de dados. Ele utiliza uma matriz de afinidade para calcular os pontos mais representativos (exemplares) para cada cluster. O Quadro 4.26, a seguir, apresenta as principais funções relacionadas a este agrupador.

Quadro 4.26: Funções de Agrupamento Affinity Propagation

Nome do método	Descrição
AffinityPropagation	Método para construção de um agrupamento de propagação de afinidade, que é um algoritmo de agrupamento que identifica clusters de dados semelhantes.
AffinityPropagationCV	Método para construção de um agrupamento de propagação de afinidade com validação cruzada, que é um algoritmo de agrupamento que identifica clusters de dados semelhantes, usando validação cruzada para escolher o melhor valor do parâmetro de número de clusters.

4.1.3.8 Gaussian Mixture Model (GMM)

O modelo de mistura gaussiana é um modelo probabilístico que tem o papel de assumir que os dados são gerados a partir de uma mistura de várias distribuições gaussianas. Ele é frequentemente usado para modelar dados onde os clusters podem não ser necessariamente esféricos. O Quadro 4.27, a seguir, apresenta as principais funções relacionadas a este agrupador.

Quadro 4.27: Funções de Agrupamento Gaussian Mixture Model

Nome do método	Descrição
GaussianMixture	Método para construção de um modelo de mistura de gaussianos, que é um modelo de aprendizado de máquina não supervisionado que usa uma mistura de distribuições normais para representar os dados.
GaussianMixtureCV	Método para construção de um modelo de mistura de gaussianos para agrupamento com validação cruzada, que é um modelo de aprendizado de máquina não supervisionado que usa uma mistura de distribuições normais para representar os dados, usando validação cruzada para escolher o melhor valor do número de componentes.

4.1.3.9 Mini-Batch K-Means

O Mini-Batch K-Means é uma versão eficiente do algoritmo K-Means que se utiliza de mini-lotes de dados em vez de todo o conjunto de dados para acelerar o processo de agrupamento. O Quadro 4.28, a seguir, apresenta as principais funções relacionadas a este agrupador.

Quadro 4.28: Funções de Agrupamento Mini-Batch K-Means

Nome do método	Descrição
MiniBatchKMeans	Método para construção de um modelo de agrupamento k-means para problemas de agrupamento com minibatching, que é um algoritmo de aprendizado de máquina não supervisionado que divide os dados em k grupos usando minibatching.

4.1.3.10 Ordering Points To Identify The Clustering Structure (OPTICS)

O algoritmo OPTICS é uma variação do DBSCAN que não requer a definição prévia do valor de densidade. Ele cria um gráfico de alcance que representa a densidade dos pontos de dados e permite a identificação de clusters em diferentes escalas de densidade. O Quadro 4.29, a seguir, apresenta as principais funções relacionadas a este agrupador.

Quadro 4.29: Funções de Agrupamento Optics

Nome do método	Descrição
OPTICS	Método para construção de um algoritmo de agrupamento baseado em densidade, que é um algoritmo de aprendizado de máquina não supervisionado que agrupa dados em base à sua densidade.
OPTICSCV	Método para construção de um algoritmo de agrupamento baseado em densidade com validação cruzada, que é um algoritmo de aprendizado de máquina não supervisionado que agrupa dados em base à sua densidade, usando validação cruzada para escolher o melhor valor do parâmetro de limiar.

4.1.4 Regras de Associação

O Scikit-learn não possui implementações nativas de algoritmos de regra de associação como o Apriori ou o Eclat. O Scikit-learn é mais focado em aprendizado supervisionado e não fornece suporte direto para mineração de regras de associação.

4.1.5 Considerações sobre a Biblioteca

Em termos de biblioteca, o scikit-learn é uma escolha sólida para muitos problemas de ML, especialmente quando é tratado tarefas tradicionais de aprendizado supervisionado e não supervisionado. A biblioteca é mantida por uma comunidade ativa de desenvolvedores *Python* e é amplamente utilizada na indústria e em pesquisas, o que significa recursos online facilmente referentes ao seu uso.

A vantagem que destaca o *scikit-learn* de outras bibliotecas, é a simplicidade e a facilidade de uso. Pois ao oferecer uma API consistente e intuitiva que facilita o desenvolvimento de modelos de machine learning, mesmo para iniciantes; em conjunto a grande variedade de algoritmos de ML, desde regressão linear e árvores de decisão até máquinas de vetores de suporte (SVM), redes neurais e algoritmos de aprendizado não supervisionado.

Embora seja excelente para experimentação e protótipos pequenos, a escolha de scikit-learn se torna limitado pois não é a melhor escolha para lidar com *datasets* muitos grandes ou modelos de *deep learning* complexos e avançados, trazendo então uma escalabilidade limitada. *Framework* como *TensorFlow* e biblioteca como *PyTorch* tendem a ser mais adequados.

4.2 PYTORCH

Em análise a documentação da própria biblioteca, nota-se que para classificação, a biblioteca possui diversos algoritmos pré-moldados, assim facilitando que seja aplicado durante a programação em Python. Comumente é utilizado para

para processo de cálculo de tensores, aceleração de GPU e diferenciação automática.

4.2.1 Cálculo de Tensores

Pytorch é capacitado para realizar cálculos utilizando tensores. Tensores são interpretados como vetores indexados (*arrays)* multidimensionais como base para operações avançadas. (SPADINI,2023). O conceito dos tensores, é que eles podem possuir qualquer número de dimensões (rank). Um tensor unidimensional é um vetor, um bidimensional é uma matriz e um tridimensional é um cubo de dados, e assim progressivamente.

Sua tipagem de dados permite que o programador especifique um tipo em específico, tal como *float, int, double etc.* Assim aprimorando o cálculo de precisão e eficiência. Tensores são aplicados na construção e treinamento de redes neurais utilizando a biblioteca *PyTorch.* Onde camadas de redes neurais e dados de entrada/saída são representados como tensores (PLAINENGLISH.IO,2021).

Assim os tensores em *PyTorch* são uma parte crucial da biblioteca e amplamente utilizados em tarefas de *ML* e processamento de dados. Pois fornecem uma flexibilidade e a eficiência recomendada para o trabalho com dados multidimensionais em uma ampla variedade de aplicações (SPADINI,2023).

4.2.2 Classificação

Pytorch é amplamente utilizado quando se trata de construção de redes neurais devido aos seus modelos classificacionais excepcionais, tais como a classificação de imagem ou redes neurais convolucionais (CNN).

4.2.2.1 RNA

Em *Pytorch,* o modelo de rede neural é criado, assim é consistido em camadas de neurônios (unidades) organizadas em uma arquitetura adequada. A

opção da arquitetura aplicada depende equivalentemente do problema a ser solucionado, mas a arquitetura simplificada inclui uma camada de entrada, uma ou mais camadas ocultas e apenas uma camada de saída (Zhang et al., 2021). O Quadro 4.30, a seguir, apresenta as principais funções relacionadas a este classificador.

Quadro 4.30: Funções de Classificação Redes Neurais Artificiais

Nome do método	Descrição
FCN	Uma rede neural feed-forward é uma rede neural que recebe um conjunto de entrada e produz uma saída. As redes neurais feed-forward são geralmente usadas para problemas de classificação e regressão.

4.2.2.2 REDES NEURAIS CONVOLUCIONAIS

Redes Neurais Convolucionais (CNNs) são amplamente usadas para tarefas que envolvem visão computacional, como classificação de imagens, detecção de objetos e pessoas e segmentação de imagens. As CNNs são uma classe de redes neurais que são particularmente eficazes na extração de recursos de imagens e na segurança da estrutura espacial dos dados (Zhang et al., 2021). O Quadro 4.31, a seguir, apresenta as principais funções relacionadas a este classificador.

Quadro 4.31: Funções de Classificação Redes Neurais Convolucionais

Nome do método	Descrição
CNN	Uma rede neural convolucional é uma rede neural que usa operações de convolução para processar dados. As redes neurais convolucionais são geralmente usadas para problemas de visão computacional, como classificação de imagens e detecção de objetos.

4.2.2.3 RESNET (REDES RESIDUAIS)

Conforme é projetado as redes neurais mais profundas e complexas, tornou-se imperativo entender como é realizado a adição de camadas que podem aumentar a complexidade e a expressividade como um todo. A capacidade de projetar redes, onde adicionar mais camadas acaba tornando-se as redes estritamente mais expressivas, ao invés que fossem diferenciadas entre si. (Zhang et al., 2021).

A ideia central é a introdução de conexões residuais onde pulam uma ou diversas camadas intermediárias na rede neural. Assim, conexões residuais permitem que os gradientes fluam de forma mais efetiva durante todo o treinamento, facilitando o treinamento de redes muito profundas. Em uma camada convolucional típica, a entrada é inserida por uma série de camadas, e a saída da camada anterior é adicionada a uma saída de camada subsequente. A tal adição é a “conexão residual”.

Em resumo, as ResNets são uma classe arquitetônica de rede neural que solucionaram o problema persistente no treinamento profundo de redes neurais, permitindo que elas sejam cada vez mais profundas, e ao mesmo tempo, mais fáceis de serem otimizadas. Elas são amplamente usadas em tarefas que demandam a visão computacional, como classificação de imagens, detecção de objetos e segmentação de imagem, e têm contribuído significativamente para o avanço do campo.

O Quadro 4.32, a seguir, apresenta as principais funções relacionadas a este classificador.

Quadro 4.32: Funções de Classificação ResNet

Nome do método	Descrição
ResNet	As redes neurais residuais são um tipo de rede neural profunda que usa uma estrutura de bloco de convolução para lidar com a dificuldade de treinamento de redes neurais profundas.

4.2.2 Agrupamento

O *PyTorch* não fornece algoritmos de agrupamento embutidos nativamente, porém, é possível a implementação de vários algoritmos utilizando as funções existentes para outras modalidades.

4.2.2.1 K-Means

A implementação do *K-Means* é de complexidade avançada, porém ao utilizar-se de loops e tensores para que possa atualizar os k-centróides iterativamente.

Os k-centróides são pontos representativos em um espaço determinado de dados dentro de um contexto de algoritmo de agrupamento, como o *K-Means*. De forma que se tornam o modelo central de cada *cluster* e são utilizados para definir a posição central de um grupo de pontos de dados semelhantes. Sendo que cada *cluster* deve possuir seu próprio k-centróide. (ALTERYX, 2023).

A ideia por trás do *K-Means* em *PyTorch* é iterativamente ajustar os k-centróides para minimizar a soma das distâncias quadradas entre os pontos de dados e seus k-centróides atribuídos, o que é chamado de inércia ou soma dos quadrados intra-cluster.

O Quadro 4.32, a seguir, apresenta as principais funções relacionadas a este agrupador.

Quadro 4.32: Funções de Agrupamento K-Means

Nome do método	Função
KMeans	Um algoritmo de agrupamento funciona iterativamente, atribuindo cada ponto de dados ao grupo mais próximo do centróide do grupo. O centróide do grupo é calculado como a média dos pontos de dados no grupo. O processo é repetido até que os centróides dos grupos não mudem mais.

4.2.2.2 DBSCAN

O algoritmo DBSCAN em *PyTorch* é baseado na densidade e requer implementações personalizadas. Onde o programador possui a liberdade de calcular a matriz de distâncias entre os pontos e implementar as regras de expansão de *cluster* e a atribuição de pontos a eles.

O algoritmo não está diretamente disponível no *PyTorch* como parte da biblioteca nativa. Porém é possível a implementação de forma manual, porém possui uma complexidade avançada, pois envolve lógica de densidade e busca de vizinhança, sendo ela um método de busca local que explora o espaço de soluções a partir de uma solução inicial. (Varun, 2021).

O Quadro 4.34, a seguir, apresenta as principais funções relacionadas a este agrupador.

Quadro 4.34: Funções de Agrupamento DBSCAN

Nome do método	Função
DBSCAN	Um algoritmo de agrupamento de densidade baseada em densidade
DBSCANGPU	Um algoritmo de agrupamento de densidade baseada em densidade em GPU

4.2.2.3 AGGLOMERATIVE CLUSTERING

O algoritmo de *clustering* aglomerativo começa com cada ponto sendo seu próprio cluster e mesclando iterativamente os *clusters* mais próximos. Assim, é utilizado os tensores *PyTorch* para calcular as distâncias e implementar a lógica de mesclagem de *cluster.*

O algoritmo não está diretamente disponível no *PyTorch* como parte da biblioteca nativa. Porém é possível a implementação de forma manual, porém possui uma complexidade avançada, pois envolve a utilização de distâncias Euclidianas para a mesclagem de *clusters*. (Varun, 2021)

O Quadro 4.35, a seguir, apresenta as principais funções relacionadas a este agrupador.

Quadro 4.35: Funções de Agrupamento Agglomerative Clustering

Nome do método	Função
AgglomerativeClustering	Um algoritmo de agrupamento hierárquico que usa um linkage para combinar os clusters. Ele usa um linkage para combinar os clusters. O linkage é uma função que calcula a distância entre dois clusters.
WardLinkage	Um linkage que combina os clusters com a menor soma dos quadrados dentro do cluster
SingleLinkage	Um linkage que combina os clusters com a distância mais curta entre os pontos mais próximos
CompleteLinkage	Um linkage que combina os clusters com a distância mais longa entre os pontos mais distantes
AverageLinkage	Um linkage que combina os clusters com a média das distâncias entre todos os pontos

4.2.3 Regressão Linear

O PyTorch não possui algoritmos de regressão linear incorporados em sua biblioteca padrão. No entanto, é fornecido as ferramentas necessárias para implementar modelos de regressão linear personalizados.

A classe *nn.Linear* permite criar uma camada linear em seu modelo, que é a base da regressão linear. Ele realiza uma transformação linear dos recursos de entrada para produzir saídas de regressão.

A perda mais comum para regressão linear é o Erro Quadrático Médio (*Mean Squared Error*). Você pode usar *nn.MSELoss* para calcular a perda entre as previsões e os rótulos.

É possível a escolha de um otimizador, como o Gradiente Descendente Estocástico (SGD) ou Adam, para atualizar os pesos do modelo durante o treinamento.

A implementação para um laço de treinamento que itere sobre seus dados de treinamento, calcule as previsões do modelo, calcule a perda e, em seguida, execute a retropropagação (backpropagation) para ajustar os parâmetros do modelo; é válida para que o desenvolvimento seja aprimorado de forma conjunto

4.2.4 Regras De Associação

PyTorch não inclui implementações específicas de algoritmos de regras de associação em sua biblioteca padrão. Algoritmos de regras de associação, como o Apriori e o Eclat, geralmente não são implementados diretamente no PyTorch. Em vez disso, esses algoritmos são mais frequentemente encontrados em bibliotecas de mineração de dados ou aprendizado de máquina, como o *Scikit-learn* ou a biblioteca *mlxtend*, que oferece implementações do Apriori e outros algoritmos de regras de associação.

4.2.5 Considerações sobre a biblioteca

O *PyTorch* traz consigo uma das características distintivas dentre as bibliotecas, que é o uso de tensores dinâmicos. Pois significa que os tensores podem ser modificados ao longo da execução do programa, o que se torna útil para construir modelos de *deep learning* com loops e estruturas de controle.

A biblioteca pode ser combinada com outras do mesmo estilo, como *OpenCV* e *scikit-learn*, podendo assim criar pipelines de ML completos, pois onde haverá defasagem de uma biblioteca a outra compensará, criando assim um fluxo benéfico ao desenvolvedor.

A biblioteca é uma escolha poderosa e flexível para tarefas de machine learning e deep learning. No entanto, a escolha entre *PyTorch* e outras bibliotecas e/ou *frameworks*, como *TensorFlow* e *PyCaret*, acaba dependendo das necessidades específicas da implementação e de preferências pessoais do desenvolvedor. Ambas as bibliotecas têm suas próprias vantagens e desvantagens, e a escolha geralmente se resume ao que melhor se adapta ao seu projeto e estilo de desenvolvimento.

5 CONCLUSÃO

As bibliotecas de ML em *Python* desempenham um papel fundamental no avanço da inteligência artificial e na resolução de uma variedade de problemas complexos em ciência de dados. Com bibliotecas como *PyCaret*, *scikit-learn*, *Keras*, *PyTorch* e muitas outras, os desenvolvedores e cientistas possuem acesso a ferramentas poderosas e flexíveis que simplificam o processo de construção e aplicação de modelos de machine learning.

Uma das maiores vantagens dessas bibliotecas são as comunidades de código aberto que as sustentam, seja em plataformas de controle de projetos como *GitHub* ou de solucionamento de dúvidas em si, tal como *Stack Overflow*. Isso resulta em uma riqueza de recursos, tutoriais e documentações disponíveis, tornando o aprendizado e a implementação mais acessíveis a todos.

Além disso, a natureza modular dessas bibliotecas permite que os desenvolvedores escolham as ferramentas que melhor se adequam às suas necessidades específicas, desde aprendizado supervisionado e não supervisionado até redes neurais profundas e processamento de linguagem natural.

O uso de bibliotecas em conjunto é uma prática comum, no ramo da programação, pois cada uma dessas bibliotecas desempenha funções diferentes em tarefas de ML e *Deep Learning*.

O *scikit-learn* é mais voltado para tarefas de ML tradicionais, como classificação, regressão e agrupamento, com ênfase em algoritmos de aprendizado supervisionado e não supervisionado. *PyTorch*, por outro lado, é uma biblioteca popular para deep learning e é altamente flexível, permitindo a criação e treinamento de redes neurais profundas. O uso de ambas juntas permite que o desenvolvedor aproveite as vantagens de ambas as bibliotecas, combinando o poder do *deep learning* com a facilidade de uso do *scikit-learn*.

O uso conjunto do *scikit-learn* e *PyTorch* pode ser altamente benéfico ao trabalhar em projetos de machine learning e deep learning. O *scikit-learn* fornece ferramentas para pré-processamento de dados, validação de modelo e algoritmos de

ML, enquanto *PyTorch* oferece flexibilidade e potência para treinar modelos de *deep learning.*

No entanto, é importante destacar que o uso eficaz de bibliotecas de ML requer uma compreensão sólida dos conceitos subjacentes e das melhores práticas de lógica de programação. O desenvolvedor deve ser capaz de escolher o algoritmo adequado, ajustar hiperparâmetros, lidar com problemas de sobreajuste e subajuste e lidar com conjuntos de dados desequilibrados.

No entanto, o sucesso depende da combinação de ferramentas sólidas com um entendimento aprofundado dos princípios de machine learning e da aplicação de boas práticas de engenharia de software.

REFERÊNCIAS

ALTERYX, **Site Alteryx**. Disponível em: <https:// https://www.alteryx.com/> Acesso em: 21/09/2023.

BHATIA, J.; **The Search for the Fastest Keras Deep Learning Backend**, 2017.

BIANCHI, A. L. **As classificações dos algoritmos de Machine Learning**. 2020.

BISHOP, C. M. **PATTERN RECOGNITION AND MACHINE LEARNING**, 2006

BORGES, L; **Python Para Desenvolvedores: Aborda Python 3.3**,2014

BOUREZ, C. **Deep Learning with Theano**. Birmingham: Packt Publishing, 2017.

BREIMAN, L. **Random forests. Machine Learning**, 2001

CARMO, A N. **Bibliotecas swing e javafx para softwares java em ambiente desktop – uma análise de usabilidade e código-fonte,** 2017.

CAROLINE, R.; RIBEIRO, A. **Ferramenta web para análise e classificação de reviews de usuários**, 2021.

CASSIANO, K. **Análise de Séries Temporais Usando Análise Espectral Singular (SSA) e Clusterização de Suas Componentes Baseada em Densidade,** 2014.

CHAVES, R. **Redes neurais deep learning aplicadas ao reconhecimento facial**, 2018

DEVMEDIA, Site Dev Media. Disponível em: <https://devmedia.com> Acesso em: 13/10/2023.

FAYAD, M. E.; SCHMIDT, D. C.; JOHNSON, R. E. **Building Application Frameworks: Object-Oriented Foundations of Framework Design**,1999.

FERREIRA, L. D. **Técnicas de aprendizado de máquina aplicadas à identificação de perfis de aprendizado em um ambiente real de ensino**. 2016.

FONTANA, E. **Introdução aos Algoritmos de Aprendizagem Supervisionada***, 2020.*

FRUTUOSO, D; **Recuperação de informação e classificação de entidades organizacionais em textos não estruturados**, 2014.

GÉRON, Aurélien. **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow**, 2022.

HASTIE, T. **Boosting,** 2003.

Hruschka, E. R., Ebecken, N. F. F. A. **Genetic algorithm for cluster analysis. IEEE Transactions on Evolutionary Computation**, 2001.

JACKSON, P.; MOULINIER, I. **Natural Language Processing for Online Applications:Text retrieval, extraction and categorization**, 2007.

KAVIANI, P.; DHOTRE, S. **Short survey on naive bayes algorithm. International Journal of Advance Research in Computer Science and Management**,2017

KHAIDEM, L.; SAHA, S.; DEY, S. R.; **Predicting the direction of stock market prices using random forest**, 2016.

LABOISSIERE, L. A.; FERNANDES, R. A.; LAGE, G. G. **Maximum and minimum stock price forecasting of Brazilian power distribution companies based on artificial neural networks. Applied Soft Computing Journal**, 2015.

LIMA NETO, G; **Machine Learning**, 2022.

LIRA, LARISSA ET AL; **Análise do algoritmo naive bayes na classificação de amostras do banco de dados hepatite**, 2023.

LOPES, G; **Aprendizado de máquina para desenvolvimento de modelo semi-empírico de predição de ruído de motor aeronáutico**, 2023.

Luxburg,U ; Schölkopf, B; **Statistical Learning Theory: Models, Concepts, and Results**,2008

MOEZ, A; **DOCUMENTATION OF PYCARET**, 2020.

Nunes, D E; **Sistema para classificação de malha viária baseado em smartphones por meio de aprendizado supervisionado**,2018.

PADILHA, V. A.; CARVALHO, A. C. P. de L.; **Mineração de Dados em Python**. ,2017

PLAINENGLISH.IO, **Pytorch - Tutorial a quick guide for new learners, 2021**

RASCHKA, Sebastian; MIRJALILI, Vahid. **Python Machine Learning**: **Machine Learning and Deep Learning with Python, scikitlearn, and TensorFlow**, 2017

Rodrigues P R S , Binda V C ; **Introdução ao uso de bibliotecas Python voltadas para criptografia de dados**,2019.

RIBEIRO, G de O. **Learning orchestra: building machine learning workflows on scalable containers**. ,2021.

Ribeiro, R;**Ferramenta web para análise e classificação de reviews de usuários de aplicativos**, 2021

RUSSELL, P. **artificial intelligence: a modern approach, global edition,** 2021.

SCALCO, F. **VISUALIZAÇÃO DE DADOS EM PROCESSOS DE MACHINE LEARNING**, 2021.

SILVA G, **Mineração de Regras de Associação Aplicada a Dados da Secretaria Municipal de Saúde de Londrina**,2004

SOUSA, M. **Modelo de Regressão Não Linear em Python**, 2022.

SPADINI, A. **Primeiros passos com Pytorch**, 2023.

STEVENS, E. et al. **Deep learning with PyTorch**, 2020.

TAVARES, C. B.; KONO, D. F. **Knowledge discovery applied to election data**, 2007.

VAN ROSSUM, G.; DRAKE JUNIOR, F. L. **Python tutorial,** 1995**.**

VARUN, Y. **HappyWhale: DBSCAN + EfficientNet baseline**, 2021.

Vasconcelos; Carvalho, **Aplicação de Regras de Associação para Mineração de Dados na Web**, 2014

VIEIRA JUNIOR, V. **Classificação de dados usando técnicas de datamining e aprendizado de máquina,** 2013

ZAVAGLIA COELHO, Alexandre; KLAFKE, Guilherme Forma; MAITO, Deíse Camargo; LATINI, Lucas Maldonado Diz; MARUCA, Giuliana; CHOW, Beatriz Graziano; FEFERBAUM, Marina. **Governança da Inteligência Artificial em Organizações: Framework para Comitês de Ética em IA**, 2023.

Zeviani et al, **Modelos de regressão não linear,** 2013

ZHANG, A.; LILPTON, Z. C.; LI, M.; SMOLLA, A. J, **Dive into Deep Learning,** 2001**.** Disponível em: <https://d2l.ai/index.html> Acesso em: 30/09/2023.