UNIVERSIDADE VEIGA DE ALMEIDA – UVA

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

COMPARAÇÃO DE DESEMPENHO ENTRE ALGORITMOS DE MACHINE LEARNING

Alexandre de Souza Pereira

RIO DE JANEIRO

2021

UNIVERSIDADE VEIGA DE ALMEIDA – UVA

Alexandre de Souza Pereira

Monografia apresentada ao curso de Ciência da Computação da Universidade Veiga de Almeida, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador(a): Carlos Alberto Alves Lemos, DSc.

COMPARAÇÃO DE DESEMPENHO ENTRE ALGORITMOS DE MACHINE LEARNING

RIO DE JANEIRO

2021

UNIVERSIDADE VEIGA DE ALMEIDA - UVA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Alexandre de Souza Pereira

COMPARAÇÃO DE DESEMPENHO ENTRE ALGORITMOS DE MACHINE LEARNING

Monografia apresentada como requisito parcial à conclusão do curso em Bacharel em Ciência da Computação.

APROVADA EM:

CONCEITO: _____

BANCA EXAMINADORA:

_____

PROF. DSc Carlos Alberto Alves Lemos, **DSc**

_____

PROF. DSc Miguel Angelo Z. de Figueiredo, **DSc**

_____

PROF. MSc Thiago Alberto Ramos Gabriel, **MSc**

Coordenação de Ciência da Computação

Rio de Janeiro

Dedico este trabalho aos meus pais, é a representação de anos de trabalho e esforço

AGRADECIMENTOS

Gostaria de agradecer todas as pessoas que passaram pela minha vida, sendo pessoal, acadêmica e profissional. Foi um longo período de aprendizado até esse momento, esse trabalho é reflexo de anos de estudo e dedicação. Eu não teria conseguido chegar até aqui sem a ajuda dos meus pais, um grupo seleto de amigos da faculdade e meus professores, que sempre tentaram passar as experiência acadêmicas e profissionais nesses últimos anos de convivência. O meu mais sincero obrigado e amo todos vocês.

RESUMO

O aprendizado de máquina ou (ML) é um ramo da inteligência computacional que explora o estudo e a construção de algoritmos baseados no aprendizado de dados em vez de instruções pré-programadas. O principal objetivo de um modelo de ML é construir um sistema de computador que aprenda a partir de um banco de dados predefinido e, eventualmente, gere um modelo de previsão, classificação ou detecção.

A utilização do ML pode ser encontrada em diversos lugares, como por exemplo na área de logística de uma empresa, em sistema de fraude em operadores de cartão de crédito ou até mesmo no Netflix ao sugestão de próximo filmes. A base de dados utilizada neste trabalho é construída com dados de qualidade dos distritos de Beijing, com dados do tipo de quantidade de poluente ,que podem ser respirados, em diâmetros de PM_{10},$PM_{2.5}$ e outras concentrações de substâncias como SO_2, CO, O_3 entre outras substâncias.

O presente trabalho teve como objetivo fazer a aplicação e análise de desempenho de diferentes algoritmos de aprendizado de máquina, sendo eles a Rede Neural Artificial, Árvore de Decisão, Floresta Aleatória, Regressão Logística, K-vizinhos mais próximos, Máquina de Vetores de Suporte. A base de dados utilizada para esse trabalho foi referente a qualidade do ar e as métrica para avaliação foram o F1_Score, Recall, Precision e matriz de confusão. As formas de treinamentos dos modelos foram separadas em algumas configurações e o modelo que apresentou o melhor desempenho dentre todos os testados foi o algoritmo de Floreta Aleatória

Palavras-Chave: Floresta Aleatória, Algoritmo de Aprendizado de Máquina, F1_Score, análise de desempenho

ABSTRACT

Machine learning or (ML) is a branch of computational intelligence that explores the study and construction of algorithms based on learning from data rather than pre-programmed instructions. The main goal of an ML model is to build a computer system that learns from a predefined database and eventually generates a prediction, classification, or detection model.

The use of ML can be found in many places, such as in the logistics area of a company, in a fraud system in credit card operators or even in Netflix when suggesting upcoming movies. The database used in this work is built with quality data from the districts of Beijing, with data on the type and quantity of pollutants that can be breathed, in diameters of PM_{10}, $PM_{2.5}$ and other concentrations of substances such as SO_2, CO, O_3, among others.

The objective of this work was to apply and analyze the performance of different machine learning algorithms, such as the Artificial Neural Network, Decision Tree, Random Forest, Logistic Regression, K nearest neighbor, Support Vector Machine. The database used for this work was related to air quality and the metrics for evaluation were the F1_Score, Recall, Precision and confusion matrix. The training forms of the models were separated in some configurations and the model that presented the best performance among all tested was the Random Flowering algorithm

Keywords: Random Forest, Machine Learning Algorithm, F1_Score, performance analysis

Sumário

1. INTRODUÇÃO

Na presente monografia, tem como objetivo principal fazer uma comparação entre as aplicações de modelos de aprendizado de máquina que melhor se encaixam com a base de dados referentes a qualidade de ar de distritos de Beijing. O aprendizado de máquina visa fazer o reconhecimento de padrões dentro da base de dados. Porém antes de abordar o assunto dos modelos de aprendizado de máquina, pontos importantes a ressaltar.

O projeto de análise de dados potencialmente utilizado para diversos recursos, podendo agregar valor de mercado nos setores de uma companhia, não só em tecnologia quanto diversos setores como em recursos humanos, logística, vendas, compras, marketing e outros. Empresas conhecidas do nosso mercado costumam utilizar essa tecnologia de dados, para fazer diversas experiências para futuras aplicações ou serviços, como por exemplo Coca-Cola, Tim, Sodexo, Santander e entre outras.

A base de dados escolhida para esta monografia, consiste em dados referentes a qualidade do ar nos distritos de Beijing, referenciadas do ano de 2013 a 2017. Inicialmente em 2013 quando os indicadores poluentes atingiram valores extremos, a OMS (Organização Mundial Saúde), declarou que Beijing se encontrava alto periculosidade a vida humana, sendo assim chamado de alerta laranja. Posteriormente, no ano de 2015, Beijing alcançou a marca de maior emissor de dióxido de carbono (CO2) e assim assumindo o alerta vermelho segundo a OMS.

No entanto os indicadores de emissões de poluição estão ligados diretamente com o crescimento econômico do país. Relatórios feitos pelo governo chinês apontam que os impactos da COVID19 para a produção podem ser observados ao analisar os mesmo. "Para combater a poluição do ar por meio da reestruturação econômica se tornarão ainda mais importantes, já que o país tem como meta de zero emissões líquidas de carbono até 2060" Segundo Lauri Myllyvirta, Analista-chefe do CREA e coautor do relatório de poluentes utilizado pelo governo chinês. [GONÇALVES, 2020].

Processo de análise é constituído de grandes bases de dados de múltiplas variáveis, que envolve diversas fontes disponíveis, podendo ser planilhas, documentos, plataformas de gestão empresarial, dessa forma essa base pode ser chamada de *Big Data.*

Iniciando o projeto de análise de dados, é possível utilizar a linguagem de programação Python para o desenvolvimento de todo o projeto em questão. É necessário fazer uma prática chamada de imersão ou exploração dos dados, que consiste em pesquisar sobre o tema, analisar quais colunas estão descritas nas bases de dados e quais tipos de informação pode-se inferir a partir disto.

Há bibliotecas que facilitam a manipulação dos dados, uma vez que essas bases são muito grandes e é necessário fazer algumas manipulações para o tratamento dos dados. Visto que, o tratamento de dados é uma prática muito necessária, porque com base nas informações adquiridas na imersão dos dados é importante que consigamos identificar os *outliers* que podem expressar dados com uma alta disparidade ou pontos como ausência de dados.

1.1 OBJETIVO.

Essa monografia, que tem como objetivo fazer a apresentação e comparação entre algoritmos relacionados ao aprendizado de máquina utilizando a linguagem Python. Para que das soluções exploradas seja possível fazer o reconhecimento de padrões das bases de dados. Sendo assim, podemos aplicar algoritmos de aprendizado de máquina, para que as tomadas de decisão do mundo real tenham um grau assertivo maior.

2. REVISÃO BIBLIOGRÁFICA.

A revisão bibliográfica foi iniciada pela pesquisa de artigos acadêmicos no SciELO, selecionando artigo e trabalhos com palavras-chave*: machine learning, deep learning, random forest, artificial neural networks.*

Para iniciar o processo de seleção dos artigos que iriam compor a revisão bibliográfica do presente monografia, analisando por parte do resumo e tipo do estudo abordado no artigo acadêmico.

Na Tabela 01 é explicitado o resultado da pesquisa bibliográfica, categorizador por utilização de algoritmo no seu desenvolvimento dos respectivos artigos científicos.

Nome	Autores	Palavras-Chave	Ano
Machine Learning na Medicina: Revisão e Aplicabilidade	Gabriela Miana de Mattos Paixão, Bruno Campos Santos, Rodrigo Martins de Araujo, Manoel Horta Ribeiro, Jermana Lopes de Moraes, Antonio L. Ribeiro	Aprendizado de Máquina, Medicina, Cardiologia	2022
Determining the geographical origin of lettuce with data mining applied to micronutrients and soil properties	Camila Maione, Eloá Moura Araujo, Sabrina Novaes dos Santos-Araujo, Alexys Giorgia Friol Boim, Rommel Melgaço Barbosa, Luís Reynaldo Ferracciú Alleoni	ICP–OES; traceability; tropical soils; heavy metals; feature selection	2021
Research on food safety sampling inspection system based on deep learning	Tzu-Chia, CHENShu-Yan YU	image processing; deep learning; network architectures; learning algorithms	2021
Ultimate Bending Strength Evaluation of MVFT Composite Girder by using Finite Element Method and Machine Learning Regressors	Zhihua Xiong, Jiawen Li, Houda Zhu, Xuyao Liu, Zhuoxi Liang	MVFT girder; ultimate bending strength; artificial neural networks; composite dowel; failure mode; LSSVM	2022
DETERMINATION OF QUALITY AND RIPENING STAGES OF ‘PACOVAN’ BANANAS USING VIS-NIR SPECTROSCOPY AND MACHINE LEARNING	Iara J. S. Ferreira, Sarah L. F. de O. Almeida, Acácio Figueiredo Neto, Daniel dos Santos Costa	quality attributes, non-destructive method, Musa spp	2022

Nome	Autores	Palavras-Chave	Ano
Forest yield prediction under different climate change scenarios using data intelligent models in Pakistan	A. Yousafzai, W. Manzoor, G. Raza, T. Mahmood, F. Rehman, R. Hadi, S. Shah, M. Amin, A. Akhtar, S. Bashir, U. Habiba, M. Hussain	climate change; forest yield; RF and KRR models; prediction; Gallies forest; Abbottabad	2021
Modelagem preditiva de propriedades mecânicas em concretos reforçados com fibra de aço utilizando redes neurais artificiais	Leonária Araújo Silva, Lucas Benício Rodrigues Araújo, Ana Karoliny Lemos Bezerra, Arthur Hermont Fonseca Murta, Lucas Feitosa de Albuquerque Lima Babadopulos, Marcelo Silva Medeiros Júnior	Concreto reforçado com fibra de aço (CRFA); Propriedades mecânicas; RNA; Análise de sensibilidade; Dosagem	2022

Tabela 01 – Artigos Utilizados na Pesquisa: Artigo, Nome dos Autores e Palavras-Chave

Inicialmente, é necessário que seja escolhida a plataforma de desenvolvimento do projeto, assim optou-se pela escolha do Google *Collaboratory*, essa ferramenta tem grandes funcionalidades, sendo a conexão direta ao drive, sendo possível executar base de dados diretamente dos repositórios de referência. Esse ambiente de desenvolvimento é construído a partir de células, possibilitando um código arrumado em blocos, podem conter código ou texto e entre outras funcionalidades.

Inicializando o projeto de ciência de dados, faz-se necessário a imersão nos dados, neste momento a base de dados vem sendo observada de uma forma geral, como por exemplo quais as colunas a base possuem, quais tipos de variáveis, qual o provedor da base e como esses dados podem começar a agregar valor à organização. À medida que o projeto vai desenvolvendo inferências do mundo real são necessários para o enriquecimento e futuros resultados esperados por ele.

Sendo assim, é necessário fazer a utilização de algumas bibliotecas, como por exemplo: *Numpy*, *Pandas*, *Seaborn*, *Matplotlib*, *Plotly*, *Scikit Learn*. A parte de tratamento dos dados, é um dos passos mais demorados dos processos de desenvolvimento, pois existem muitos pontos a serem analisados, pontos como *outliers*, que são dados faltantes ou dados duvidosos. No entanto, quando se trata de aprendizado de máquina, é interessante fazer a manipulação dos dados, para que não seja necessário perder alguns dados.

À medida que os valores faltantes são encontrados, temos possibilidades de tratamento dos mesmo. Verificando qual o tipo de coluna que estamos tratando, se é coluna com variáveis numérico ou categóricas. Existem algumas formas de fazer o tratamento, sendo por meio de

substituição de valores, exclusão de valores indefinidos ou até mesmo exclusão de toda a coluna do *data frame*.

A primeira forma de tratamento, em caso de variáveis numéricas pode-se substituir os valores faltantes pela média ou mediana dos valores existentes, para que assim não se perca volume de dados e comprometa os algoritmos de aprendizado de máquina.

A segundo forma consiste em excluir as linhas do *data frame* que possuam dados faltantes, essa medida pode ser utilizada quando os valores de determinada coluna são do tipo categórico, que acaba não possibilitando a substituição de valores.

A terceira forma consiste em fazer a exclusão de toda a coluna, sendo por motivos de alto índices de dados faltantes ou a coluna não apresenta nenhum nível de correlação com o resto do *data frame*.

Na linguagem de programação *Python*, é possível fazer operações matemáticas de forma simples e rápida, tendo em mente que nestes projetos de aprendizado de máquinas manipula-se grandes bases de dados. Para que seja possível fazer a manipulação, utiliza-se a biblioteca do Pandas para fazer a manipulação da massa de dados. Assim tornando a visualização inicial mais fácil.

O *data frame* é uma estrutura que guarda diversos tipos de variáveis, no estilo de matriz. Sendo assim podendo se orientar pelos índices das respectivas colunas distribuídas nela. Ao utilizar o *data frame*, pode ser necessário fazer uso de métodos estatísticos, como por exemplo a utilização de desvio padrão, mediana, média, quartil etc.

A biblioteca Numpy é geralmente utilizada quando se julga necessário fazer manipulações em matrizes ou vetores contendo diversos tipos de dados, assim aparecendo como um facilitador da linguagem. Sendo assim, existem muitas formas de utilizar as funções do Numpy, como por exemplo utilizar estatística básica, rápidas operações entre conjunto de dados, operações algébricas, simulações aleatórias, organização de dados, seleção entre outras formas.

A biblioteca Seaborn podendo ser utilizado para a montagem das informações dispostas nas bases de dados. É possível fazer a montagem de diversos tipos de gráficos como por exemplo: gráfico de barra, círculo, linhas, histograma.

A biblioteca Scikit Learn, também conhecida por sklearn, é utilizada para executar os algoritmos de *machine learning*, pré-processamento das informações e as métricas para avaliação. Neste projeto iremos abordar os a algoritmos de classificação, os algoritmos são o *Randon Forest*, *Logistic Regression*, *Decision Tree, K-Nearest Neighbors*, *SVM*, Rede Neural Artificial (RNA Perceptron) e RNA(Multicamadas). Já as métricas escolhidas foram as *Accuracy*, *F1_Score* e a matriz de confusão.

3. ALGORITMOS DE MACHINE LEARNING.

Os modelos utilizados nesta monografia para o desenvolvimento, tendo como objetivo fazer a comparação de desempenho são algoritmos que buscam classificar informações semelhantes. Para obter os resultados dos treinamento dos modelos, será utilizado as métricas para julgar quanto de porcentagem o modelo foi assertivo, em cada uma dos treinos.

3.1 Rede Neural Artificial - (RNA):

O algoritmo de rede neural se apresenta como um algoritmo para resolução de problemas complexos, já existe algumas aplicações em nossa sociedade, como por exemplo:

- Descoberta de novos remédios.
- Carros autônomos.
- Entendimento de linguagem natural.
- Solução de controle de tráfego.
- Cura para doenças

Dentre as aplicações anteriores, não existe um algoritmo para a solução destes problemas, o RNA irá se adaptar a base de dados que ele foi exposto e partindo para uma solução. A situação mais comum para utilização desse algoritmo é em casos de problemas complexos e bases de dados extremamente grande a ponto de ser chamado de *Big Data*

Estrutura do algoritmo é inspirada em uma rede neural humana, sendo composta de neurônios, axônio, sinapses. Sendo assim, uma RNA é formada por um conjunto de neurônios, neste tópico iremos abordar os neurônios com arquitetura do *Perceptron*.

A estrutura do neurônio artificial, possui um número de entradas indeterminado, cada uma dessa entradas possui um peso e tudo está ligado a uma função de ativação, que será determinante para a ativação do neurônio, a estrutura que utilizaremos é chamada de unidade linear com threshold (LTU, do inglês), sendo a unidade mais simples e inicial. Podemos observar um exemplo na figura 01

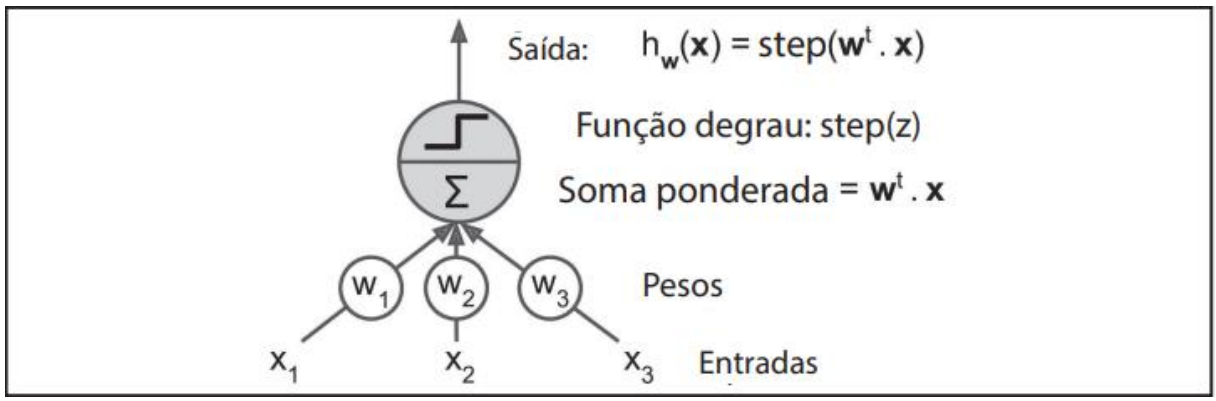

Figura 01: Unidade linear com threshold. Fonte: Mãos à Obra Aprendizado de Máquina com Scikit-Learn & TensorFlow – 17/04/2022.

Na prática, esse número de entrada são as variáveis presentes na base de dados, onde o algoritmo de RNA busca o melhor peso possível para a base em questão. Os pesos são chamados de sinapse e possuem duas classificações possíveis, se o peso é positivo é chamado de sinapse excitadora, senão ele é chamado de sinapse inibidora. Eles também são responsáveis por amplificar ou reduzir o sinal de entrada do neurônio.

Conforme o desenvolvimento das aplicações, o algoritmo foi se modificando e uma ramificação foi criada, chama de *Perceptron* Multicamada e Retropropagação (MLP), que possuía uma arquitetura diferente, visto que supria pontos fracos do algoritmo tradicional, onde na classificação lógica de OR (ou da lógica matemática) e de XOR (ou exclusivo da lógica matemática)

O algoritmo MPL utiliza uma função de ativação diferente do *Perceptron,* essa mudança de arquitetura ocorreu pela não adaptabilidade do algoritmo em algumas situações, com isso a função degrau foi substituída, podendo ter duas opções de escolha:

- Função Tangente Hiperbólica Tanh.
- Função ReLU

Em resumo, para os treinamento dos algoritmos de RNA, buscando por um melhor classificação dos pesos, o algoritmo utiliza a regra de Hebb ou aprendizado de Hebbiano, que busca fazer a atualização dos pesos das saídas que obtiveram resultados corretos, sendo assim melhorando o modelo como um todo. A próxima exemplifica a estrutura de uma MLP.

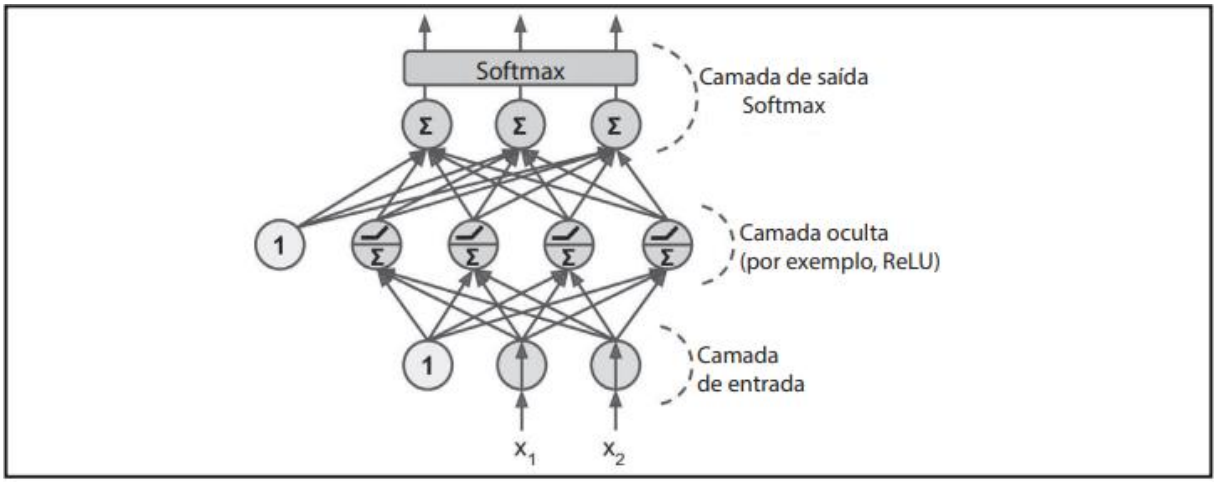

Figura 02: Modelo de Classificação MPL. Fonte: Mãos à Obra Aprendizado de Máquina com Scikit-Learn & TensorFlow – 17/04/2022.

3.2 Árvore de Decisão - (*Decision Tree*):

Classificado como um algoritmo versátil, pois é possível fazer tarefas tanto de regressão como de classificação, sendo assim poderosos, para tratar de bases de dados complexas. Esse algoritmo se torna mais atraente, visto que para utilizá-lo não é necessária uma base de dados altamente tratada ou escalonada conforme a distribuição das informações presentes. À medida que o treinamento do algoritmo vai ocorrendo, existem ferramentas de visualização de árvores na biblioteca do *Scikit-Learn*, para assim entender como as classificações do modelo estão acontecendo.

Exemplo de plotagem visual de uma árvore de decisão. Caso a *pental length* menor que 2,45cm o algoritmo irá classificar como setosa, caso contrário o algoritmo irá fazer um número X de perguntas para que assim a classificação seja feita

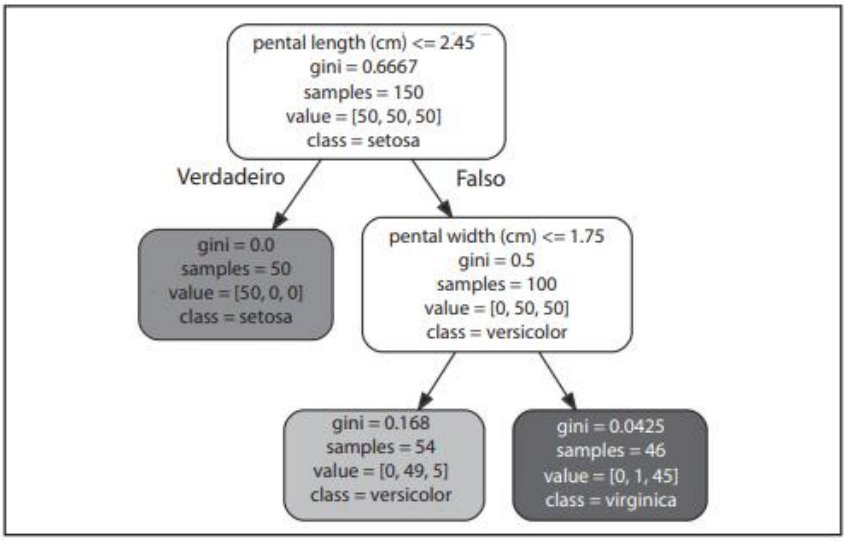

Figura 03: Árvore de Decisão da Iris. Fonte: Mãos à Obra Aprendizado de Máquina com Scikit-Learn & TensorFlow – 16/04/2022.

3.3 Floresta Aleatória – (Random Forest):

O algoritmo da Floresta Aleatória é uma evolução do algoritmo citado anteriormente da Árvore de Decisão. Esse algoritmo utiliza uma técnica chamada de *Ensemble Learning* (Aprendizado em conjunto), essa técnica consiste em fazer classificações por meio de votação da maioria, muito semelhante a votação de escolha do algoritmo do k-NN.

Ele faz a construção de N árvores, com uma classificação intermediaria que será votada por essas N árvores e a mais votada será a classe final determinada. Visto que esse algoritmo é uma ramificação da Árvore de Decisão, ele possui os mesmo hiper parâmetros e mais alguns que

representam os parâmetros da técnica de *Ensemble Learning*. Apresentando o modelo de Floresta Aleatória na figura 04, como é feito a geração das N árvores e a votação da classe,

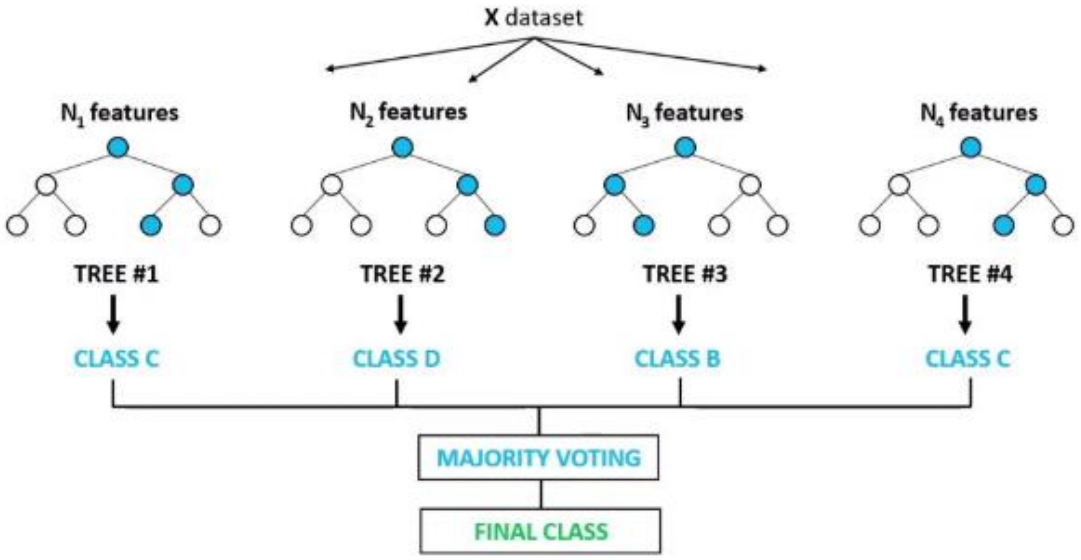

Figura 04: Random Forest Classifier. Fonte: cibersistemas.pt/tecnologia/como-usar-algoritmos-baseados-em-arvore-para-aprendizado-de-máquina/– 17/04/2022.

3.4 Regressão Logística:

A regressão logística pode ser utilizada tanto para algoritmos de regressão como também classificação. Normalmente esse modelo tem como objetivo, classificar uma ação como verdadeira, rotulada como 1, caso tenha a probabilidade estimada mais ou igual a 50%, caso contrário a ação seria falsa, rotulada como 0.

O funcionamento do modelo de regressão logística, tem como característica uma equação de reta que gera um gráfico sigmoide, ou seja, em formato de S. Como podemos observar o exemplo a seguir na figura 05. Assim o treinamento e função de custo, pode ser expresso por uma função matemática chamada de log *loss*.

$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$

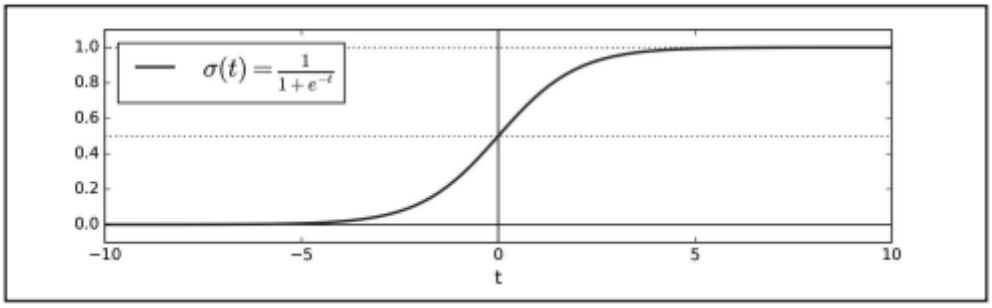

Figura 05: Função Logística. Fonte: Mãos à Obra Aprendizado de Máquina com Scikit-Learn & TensorFlow – 16/04/2022.

3.5 K-Vizinho mais próximos - (k-NN):

Esse algoritmo também pode ser chamado de KNN ou k-NN, geralmente utilizado para tratar problemas de classificação, no entanto também pode ser utilizado para regressão. Neste tópico iremos abordar para modelos referente a classificação. A forma de classificar que esse algoritmo utiliza é por meio de votação de K vizinhos mais próximos, o número de vizinho geralmente é um número ímpar e a distância pode ter algumas opções de escolha, como as distância:

- Euclidiana
- Manhattan
- Minkowski
- Hamming

Figura 06: Classificação k-NN Fonte: https://www.ibm.com/topics/knn – 10/04/2022.

3.6 MÁQUINA DE VETORES DE SUPORTE (SVM)

O algoritmo SVM, pode ter algumas variações, sendo classificação linear, não linear, de regressão e identificador de outliers. No entanto, neste tópico só irão ser abordados os algoritmos de classificação linear e não linear. Este algoritmo de classificação é apropriado para base de dados complexos, como reconhecimento de imagem ou de escrita, sendo de pequena ou média dimensão.

Exemplificando o algoritmo de SVM como apresentado na figura 07, faz-se a classificação por meio da separação de um hiperplano que é representada por pela linha continua, já a linha tracejada representa a margem se segurança que também é chamada de vetor de suporte.

Para realizar a classificação, o algoritmo gera vários hiperplanos que buscar a melhor separação entre as classes. Para que haja a melhor distinção entre elas, o SVM utiliza os vetores de suporte, de modo que a maior distância será utilizada pelo algoritmo.

Pode-se observar que na figura X02, na aplicação mais à esquerda, a margem de segurança não foi aplicada corretamente, no entanto, por mais que as classes possam vir a estar bem definidas, o vetor de suporte não consegue aplicar a distância de segurança para definição entre uma classe e outra. Já na aplicação mais à direita, as classificações da base de dados se tornam mais bem definida e o vetor de suporte representando a distância entre as classes bem determinado, possibilitando a fixação dos pontos pertencentes a cada classe. A classificação abaixo é chamada de classificação linear de margens largas.

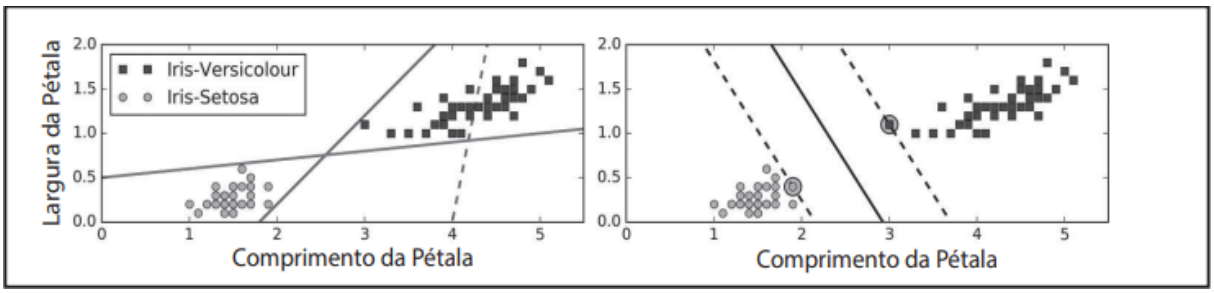

Figura 07: Algoritmo de SVM linear. Fonte: Mãos à Obra Aprendizado de Máquina com Scikit-Learn & TensorFlow – 13/04/2022.

Para casos de aplicação do SVM não linear, são casos que as classes não possível de ser separadas por hiperplano, como no linear. É necessário fazer o ajuste dos hiperparâmetros, que seria o kernel e o C. O kernel séria qual método de característica que o algoritmo mude e tenha diferentes desempenhos, já o parâmetros C é o ajuste de erro da classificação, quanto maior o valor de C mais bem separado as classes serão e com um C menor, o algoritmo fica mais permissivo a erros. Os tipos de kernel que o algoritmo pode apresentar são:

- Linear;
- Gaussiana
- Polinomial
- Tangente Hiperbólica;

Por consequência, após a mudanças dos parâmetros, a complexidade do algoritmo aumente consideravelmente, junto ao seu tempo de execução. Podemos observar um exemplo de SVM não-linear na figura 08

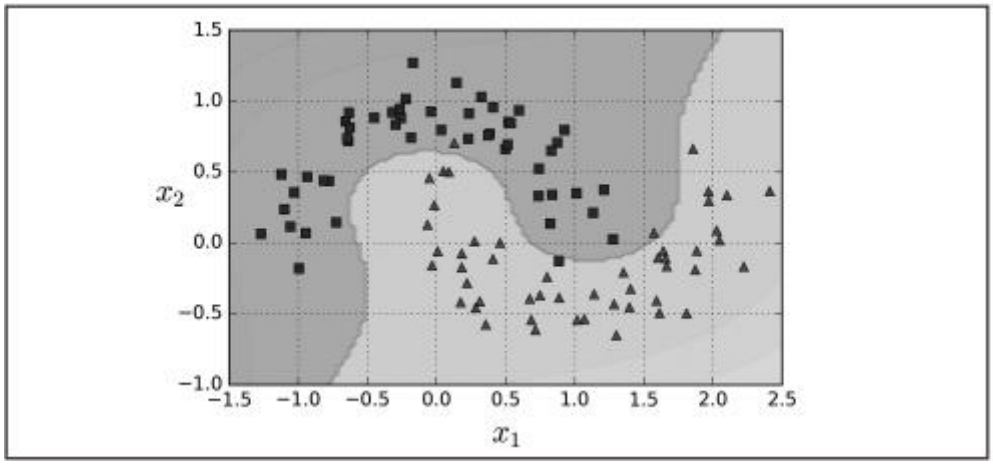

Figura 08: Algoritmo de SVM não linear. Fonte: Mãos à Obra Aprendizado de Máquina com Scikit-Learn & TensorFlow – 13/04/2022.

4. ESTUDO DE CASOS.

A base de dados escolhido, possui dados de qualidade do ar, dos distritos de Beijing, composta por dados numéricos e categóricos, como por exemplo: ano, mês, dia, hora, concentração de PM_{10}, $PM_{2.5}$, SO_2, NO_2, O_3, CO, por (ug/m^3), temperatura (°C), pressão, temperatura do ponto de orvalho (DEWP), chuva (mm), velocidade do vento (WSPM), direção do vento (wd), e o nome da estação que foi coletado.

A concentração de partículas $PM_{2.5}$ e PM_{10} são as quantidades de partículas imersas no ar, tipo de poluição que é possível se inalada. Essas quantidades são regularizadas por um *Environmental Protection Agency* (EPA), dos Estados Unidos.

	year	month	day	hour	PM2.5	PM10	SO2	NO2	CO	O3	TEMP	PRES	DEWP	RAIN	wd	WSPM	station
0	2013	3	1	0	4.0	4.0	4.0	7.0	300.0	77.0	-0.7	1023.0	-18.8	0.0	NNW	4.4	Aotizhongxin
1	2013	3	1	1	8.0	8.0	4.0	7.0	300.0	77.0	-1.1	1023.2	-18.2	0.0	N	4.7	Aotizhongxin
2	2013	3	1	2	7.0	7.0	5.0	10.0	300.0	73.0	-1.1	1023.5	-18.2	0.0	NNW	5.6	Aotizhongxin
3	2013	3	1	3	6.0	6.0	11.0	11.0	300.0	72.0	-1.4	1024.5	-19.4	0.0	NW	3.1	Aotizhongxin
4	2013	3	1	4	3.0	3.0	12.0	12.0	300.0	72.0	-2.0	1025.2	-19.5	0.0	N	2.0	Aotizhongxin
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
418941	2017	2	28	19	11.0	27.0	4.0	20.0	300.0	81.0	12.6	1011.9	-14.3	0.0	N	2.0	Wanliu
418942	2017	2	28	20	15.0	43.0	6.0	55.0	500.0	45.0	9.4	1012.3	-11.9	0.0	WSW	1.0	Wanliu
418943	2017	2	28	21	13.0	35.0	7.0	48.0	500.0	48.0	8.7	1012.8	-13.7	0.0	N	1.1	Wanliu
418944	2017	2	28	22	12.0	31.0	5.0	47.0	500.0	50.0	7.8	1012.9	-12.6	0.0	NNE	1.0	Wanliu
418945	2017	2	28	23	7.0	25.0	6.0	86.0	700.0	11.0	7.0	1012.6	-11.2	0.0	NE	1.1	Wanliu

418946 rows × 17 columns

Figura 09: Apresentação de base de dados. Fonte: Autoral – 05/09/2021.

Dentre esses atributos citados anteriormente, há algumas variações dentre os valores como valores inteiros, reais, cadeia de caracteres. Os tipos das variáveis são separados por classificações como: numérica contínua, numérica discreta, categóricas nominais, categóricas ordinais.

Visualizando de forma mais prática, as variáveis numéricas têm como característica ser do tipo número inteiro para as discretas, como por exemplo dias, meses, anos. Já para o tipo contínuo, tem a característica de número real, como por exemplo temperatura, pressão atmosférica e velocidade do vento.

Seguindo, para as variáveis categóricas, iniciando pela nominal, são dados que são mensuráveis e sem ordem, como por exemplo cor dos olhos, gênero, nome, cor de pele. Já o tipo ordinal tem como característica fazer a classificação sob uma lógica, como por exemplo tamanho P, M, G.

Originalmente a base de dados estão organizadas em tabelas do Excel, separada por região de amostras coletadas, sendo assim o próximo passo o tratamento das bases de dados.

Como foi mostrado anteriormente, a apresentação da base de dados é feita utilizando a biblioteca Pandas, com a função *head*(x) que faz a apresentação dos cinco primeiros e últimos valores. Essa função não necessariamente precisa ter um valor definido, se tiver o retorno será do *data frame* com o número definido, caso tenha retornará como na figura 09.

4.1 Tratamento e preparação dos dados.

Inicializando o procedimento de limpeza de dados, utilizando as bibliotecas de Python como Pandas e Numpy que possibilitam a visualização de dados, tratamento ou exclusão do dado. É possível fazer de algumas formas a verificações na base.

Nota-se que, o pré-processamento pode ser iniciado pela variáveis dos tipos numéricas, utilizando a função *describe()* do Pandas. Essa função descreve cada coluna da base em algumas métricas estudadas geralmente em estatísticas, como por exemplo: contagem, média, desvio-padrão, quartil, valor máximo e mínimo.

```
df.describe.transpose()
```

	count	mean	std	min	25%	50%	75%	max
No	139577.0	17492.762375	10117.477605	1.0000	8729.0	17485.0	26257.000000	35064.0
year	139577.0	2014.657888	1.176495	2013.0000	2014.0	2015.0	2016.000000	2017.0
month	139577.0	6.524986	3.445526	1.0000	4.0	7.0	10.000000	12.0
day	139577.0	15.719997	8.803012	1.0000	8.0	16.0	23.000000	31.0
hour	139577.0	11.507319	6.918404	0.0000	6.0	12.0	17.000000	23.0
PM2.5	139577.0	77.271392	77.908939	2.0000	20.0	54.0	106.000000	882.0
PM10	139577.0	103.448947	89.904361	2.0000	36.0	83.0	143.000000	999.0
SO2	139577.0	15.302150	21.391419	0.2856	2.0	7.0	18.689242	500.0
NO2	139577.0	47.652956	33.686218	1.0265	22.0	40.0	67.000000	276.0
CO	139577.0	1204.343520	1106.452394	100.0000	500.0	900.0	1400.000000	10000.0
O3	139577.0	58.647159	55.568804	0.2142	14.0	48.0	81.000000	450.0
TEMP	139577.0	13.437556	11.481842	-19.9000	3.1	14.4	23.200000	41.6
PRES	139577.0	1009.176319	10.353072	982.4000	1000.8	1008.8	1017.300000	1042.0
DEWP	139577.0	2.195378	13.878994	-35.3000	-9.4	2.6	15.000000	29.1
RAIN	139577.0	0.064257	0.808984	0.0000	0.0	0.0	0.000000	52.1
WSPM	139577.0	1.683190	1.253197	0.0000	0.9	1.3	2.100000	12.9

Figura 10: plotando descrição da base. Fonte: Autoral – 12/09/2021.

Caso as variáveis possuam valores categóricos, utiliza-se a função *isnull*(), para retornar o *data frame* com todos os valores booleanos, se existe valor retorna *true*, caso contrário false. A fim de visualizar a quantidade de valores nulos do *data frame*, atrela-se a função *sum*() junto a *isnull*(), para que assim seja retornado por cada coluna as quantidades de valores nulos do *data frame.*

Como citado anteriormente, o processo de tratamento de dados é um trecho importante quando se trata de projeto de análise de dados. Com isso a base utilizada neste projeto estava segmentada em doze partes, sendo dados histórico de 2013 a 2017 das estações descritas a seguir: Aotizhongxin, Changping, Tiantan, Wanshouxigong, Dingling, Dongsi, Gucheng, Huairou, Guanyuan, Nongzhanguan, Shunyi, Wanliu.

Antes de agrupar todas essas bases em um grande conjunto, julgou-se necessário fazer o tratamento dos dados faltantes de todas as base em separado para que assim as particularidade de cada uma das regiões não tivesse alteração. Visto que ao agrupar todas as parte as médias das tabelas ficariam diferentes por conta das particularidade de cada região do dado colhido.

Portanto o método de tratamento de dados escolhido foi o de substituição de valores inteiros e reais ausentes e exclusão de linhas. A substituição dos valores indefinidos foi feita por meio da mediana dos valores presentes na base de cada coluna respectivamente, já os valores que foram excluídos sendo do tipo categórico.

Seguindo o desenvolvimento, é necessário fazer com que a base de dados tenha somente informações que possam ser aproveitadas no processo de aprendizado de máquina, no qual serão os próximos passos. Com a necessidade de ajuste da base, utilizou-se a matriz de correlação, essa ferramenta é utilizada para verificar a relação entre as variáveis da base de dados, como pode-se observar na figura 11.

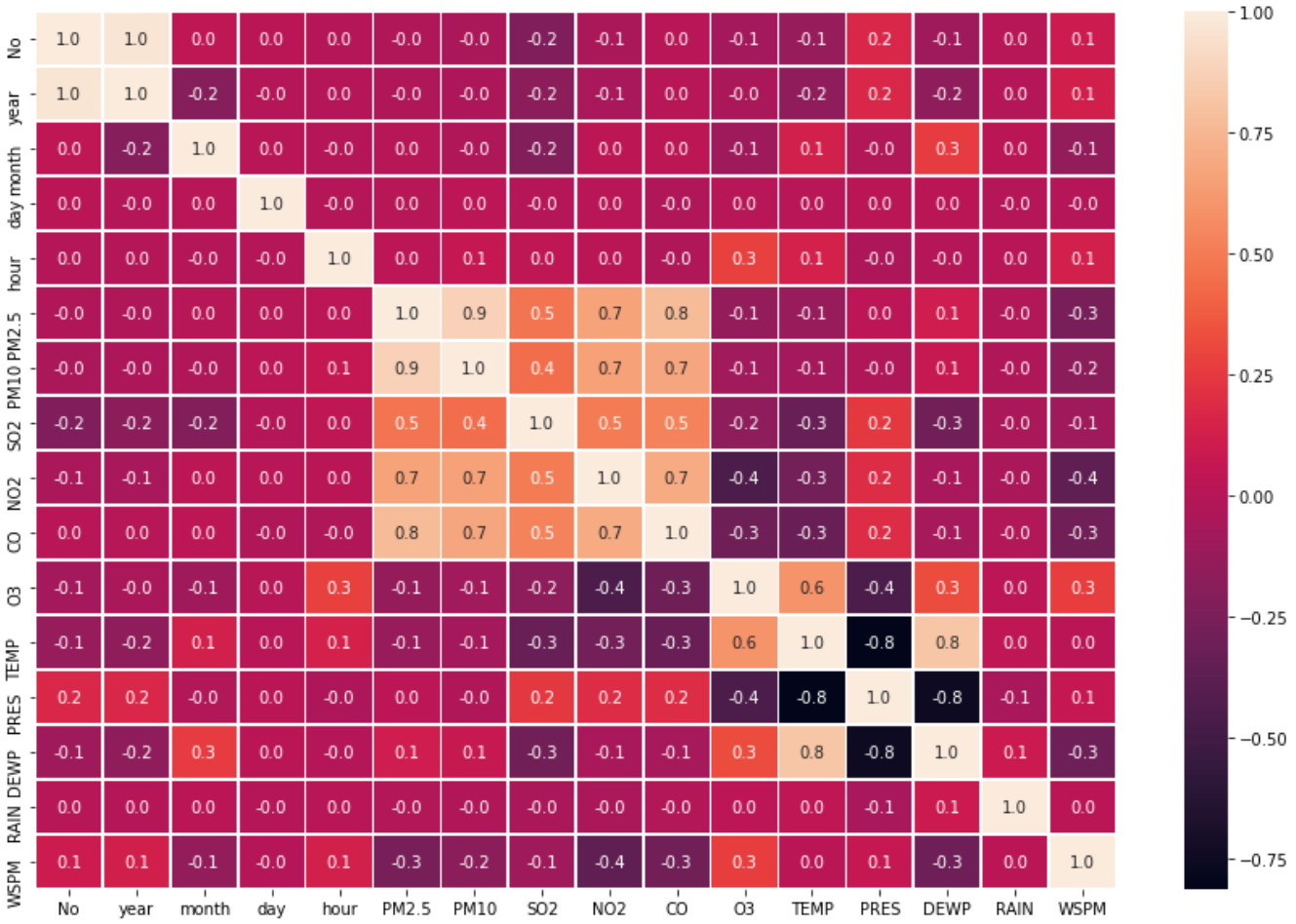

Figura 11: Matriz de correlação. Fonte: Autoral – 02/11/2021.

Ao utilizar a matriz de correlação, julgou-se necessário fazer a comparação entre as colunas para melhor adequar quando for aplicador os modelos de aprendizado de máquina. Com isso gera-se um valor das comparações fixado entre 1 e -1. No entanto, na matriz em questão os número estão no intervalo de 1 até -0,8.

Logo após a utilização da matriz de correlação, podemos observar que algumas colunas não se relacionavam entre si e podem ser excluídas do *data frame*. Portanto, não serão levadas em consideração neste trabalho as colunas: número da linha, ano, mês, dia e hora

4.2 Separação das Bases

Iniciando o processo de separação das bases de dados, sendo as base de treino e teste. Para que os modelos de aprendizado de máquina possam ser executados, as base de dados são submetidas a uma separação, como descrito anteriormente. No entanto, existe um problema que durante o treinamento, que é chamado de underfitting e overfitting.

Segundo os autores da Ian Goodfellow, Yoshua Bengio, Aaron Courville [Deep Learning (2017, MIT), Cap 5, Pag 111], “ 1. Fazer com que o erro de treino seja pequeno 2. Tornar

pequeno o intervalo entre o treino e o erro de teste. Estes dois fatores correspondem aos dois desafios centrais na aprendizagem de máquinas: o underfitting e o overfitting. O underfitting ocorre quando o modelo não é capaz de obter um valor de erro suficientemente baixo no conjunto de formação. O overfitting ocorre quando o intervalo entre o erro de treino e o erro de teste é demasiado grande. Podemos controlar se um modelo é mais susceptível de se overfit ou underfit, alterando a sua capacidade".

Os autores da situação apontam os fatores onde ocorrem os problemas, uma vez que esses problema são pontuados, é necessário tomar uma série de cuidados para que os modelos de aprendizado de máquina não fiquem tendenciosos. Conforme a separação de dados é feita, os grupos necessariamente precisam possuir características diferentes, de forma a base de teste oferecer um desafio ao modelo que está tentado estabelecer o resultado com base nos dados de treino.

Visto que os problemas citados anteriormente estão ligados diretamente com o balanceamento das quantidade de bases treino e teste, sendo assim fatores que podem estar diretamente ligados com o problema em questão. Geralmente, utiliza-se 70% e 30%, para base de treino e teste respectivamente.

Seguindo a linha de desenvolvimento e análise, algumas colunas foram retiradas do processo, por conta de não apresentarem valor quando o assunto é algoritmos de *machine learning,* sendo colunas como quantidade de chuva (Rain), direção do vento (WD) e velocidade do vento (WSPM)

4.3 Treinamentos

Dando início aos treinamento dos modelos, optou-se por fazer treinamento variando a quantidade dos dados e comparando os seus desempenhos. A seleção de bases de dados foi feita por meio de comparação da menor quantidade de valores faltantes por base, a fim de utilizar bases com poucas modificações.

Visto que a seleção foi feita, temos como resultado os *data frame*s, sendo nome da estação e quantidade de dados faltantes: Nongzhanguan - 4090 (data_10), Gucheng - 4728 (data_8), Wanshouxigong - 5146 (data_7), Changping - 5166 (data_2), Tiantan - 5277 (data_3), Guanyuan - 5279 (data_9), Wanliu - 6447 (data_12),Dingling - 7015 (data_4), Aotizhongxin - 7271 (data_1), Huairou - 7485 (data_6), Dongsi - 7600 (data_5), Shunyi - 8523 (data_11)

As configurações de treinamento foram montadas de forma que a primeira configuração irá ter duas bases de dados a data_10 e o data_8. Sempre após o treinamento e teste do *data frame*,

será salvo os resultados de *Accuracy, F1_Score, Precision Score, Recall Score* e a matriz de confusão, posteriormente adicionando mais duas bases e reproduzindo o processo até que todas as bases sejam utilizadas. A seguir temos os formatos e fórmulas das métricas utilizadas.

- Matriz de Confusão:

		Valor Real	
		Sim	Não
Valor Previsto	Sim	Verdadeiro Positivo (TP)	Falso Negativo (FN)
	Não	Falso Positivo (FP)	Verdadeiro Negativo(TN)

Tabela 02: Matriz de confusão. Fonte: Autoral – 10/05/2022.

- Accuracy:
  - A métrica da accuracy, faz a comparação entre a predição do modelo e a classe que realmente é verdadeira
- Precision:
  - TP / (TP + FP) TP*FP
- Recall:
  - TP / (TP + FN)
- F1_Score:
  - `F1_Score = 2 * (precision * recall) / (precision + recall)`

As colunas do d*ata frame* utilizados na presente monografia foram: PM_{10}, $PM_{2.5}$, SO_2, NO_2, O_3, CO, TEMP, PRES. Com relação ao número de instancias de cada base de dados utilizada é aproximadamente 34800. Além disso a base de dados foi separada entre X e y, sendo os atributos que os modelos irão processar e classe a que o dado, respectivamente.

Logo que essa separação foi feita o próximo passo foi separar as bases entre treino e teste, utilizou-se a função *train_test_split*, que por padrão ela recebe os atributos (X), as classes (y), *random_state* e o *test_size*. O *random_state* é responsável por escolher sempre a mesma faixa de dados de treino e teste, já o *test_size* é o tamanho da base de teste escolhida. Nos treinamento a configuração foi feita de forma a utilizar o *random_state* = 95 e o *test_size* = 0.3.

primeiros treinamentos foram executados com as configurações de *data frames* com a menor quantidades de dados faltantes, sendo assim os primeiros teste foram executados com seis configurações diferentes e os modelos não seus formatos padrões. As configurações de treinamento são:

- Conf_01: data8,data10
- Conf_02: data2,data7,data8,data10
- Conf_03: data2, data3, data7, data8, data9, data10

- Conf_04: data2, data3, data4, data7, data8, data9, data10, data12
- Conf_05: data1, data2, data3, data4, data6, data7, data8, data9, data10, data12
- Conf_06: data1, data2, data3, data4, data5, data6, data7, data8, data9, data10, data11, data12

Executando os primeiros treinamento sem nenhum pré-processamento dados, o tempo de execução estava relativamente alto e as taxas de assertivas baixas. Com a finalidade de melhorar o desempenho dos modelos, foi testou-se dois pré-processamento da biblioteca do *SK Learn* sendo elas o *MinMaxScaler* e o *StandardScaler*, os resultados mais satisfatórios foram encontrados com a utilização do *MinMaxScaler*.

4.4 Testes

Os resultados encontrados dos modelos com as configurações descritas acima, foram:

modelo	conf.	Precision	Recall	F1_Score	Accuracy	Tempo Treinamento
Random Forest	1	88%	88%	88%	88%	00:06:51
Logistic Regression	1	80%	80%	80%	80%	00:00:17
Decision Tree	1	80%	80%	80%	80%	00:00:31
SVM	1	83%	83%	83%	83%	00:54:32
RNA	1	78%	71%	70%	72%	00:00:05
Multi_RNA	1	85%	84%	84%	84%	00:30:58
KNN	1	86%	86%	86%	86%	00:00:29
Random Forest	2	62%	62%	61%	62%	00:18:09
Logistic Regression	2	43%	45%	43%	45%	00:03:49
Decision Tree	2	47%	47%	47%	47%	00:01:19
SVM	2	50%	50%	48%	49%	09:36:04
RNA	2	52%	29%	19%	29%	00:00:02
Multi_RNA	2	52%	52%	51%	52%	01:15:39
KNN	2	59%	58%	58%	58%	00:00:06
Random Forest	3	45%	44%	44%	44%	00:30:32
Logistic Regression	3	29%	30%	28%	30%	00:06:58
Decision Tree	3	28%	34%	31%	31%	00:01:03
SVM	3	34%	34%	34%	34%	11:34:29

modelo	conf.	Precision	Recall	F1_Score	Accuracy	Tempo Treinamento
RNA	3	45%	24%	17%	24%	00:01:34
Multi_RNA	3	35%	34%	33%	34%	02:31:11
KNN	3	37%	36%	36%	36%	00:01:24
Random Forest	4	38%	39%	38%	39%	00:46:14
Logistic Regression	4	21%	24%	20%	24%	00:11:09
Decision Tree	4	26%	26%	26%	26%	00:02:26
SVM	4	26%	28%	25%	28%	12:27:57
RNA	4	16%	13%	7%	13%	00:01:19
Multi_RNA	4	27%	28%	27%	28%	02:44:22
KNN	4	31%	31%	31%	31%	00:01:46
Random Forest	5	43%	44%	43%	44%	01:01:38
Logistic Regression	5	17%	20%	16%	19%	00:16:27
Decision Tree	5	28%	28%	28%	28%	00:02:46
SVM	5	22%	23%	20%	23%	04:44:17
RNA	5	13%	15%	11%	15%	00:01:59
Multi_RNA	5	25%	26%	25%	26%	03:00:27
KNN	5	33%	32%	32%	32%	00:02:23
Random Forest	6	42%	42%	42%	42%	01:13:07
Logistic Regression	6	16%	18%	15%	18%	00:20:02
Decision Tree	6	26%	26%	26%	26%	00:03:29
SVM	6	21%	22%	20%	22%	06:10:05
RNA	6	15%	9%	3%	9%	00:15:26
Multi_RNA	6	23%	24%	23%	24%	04:03:27
KNN	6	32%	31%	31%	31%	00:01:09

Tabela 03: Resultados dos Modelos. Fonte: Autoral – 17/05/2022.

Os modelos foram utilizados com suas versões bases, nessas condições como pode-se observar na tabela acima, os modelos de *Random Forest* e *KNN* tiveram os melhores resultado nas diferentes configurações exploradas nos treinamentos.

Posteriormente, foi utilizada para a tentativa de melhorar de desempenho dos modelos a utilização da função de *GridSearchCV*. Pois, essa função tem como objetivo a otimização de resultados encontrados, utilizando os parâmetros passados e validação cruzada, para chegar nos

melhores paramentos, retornando assim a melhor configuração possível dentro do ambiente apresentado.

Os parâmetros utilizados para cada um dos modelos, foram estudados e analisado perante a documentação do *SK Learn*. Listando a seguir os modelos e seus parâmetros testados.

- Arvore de Decisão (*Decision Tree*):
    - *'criterion': ['gini', 'entropy'],*
    - *'splitter': ['best','random'],*
    - *'min_samples_split': [2,5,10],*
    - *'min_samples_leaf': [1,5,10]*

- Floresta Aleatória (*Random Forest*):
    - *'criterion': ['gini', 'entropy'],*
    - *‘n_estimators': [10,40,100,150],*
    - *‘min_samples_split': [2,5,10],*
    - *‘min_samples_leaf': [1,5,10]*
- *KNN:*
    - *'n_neighbors': [3,5,10,20],*
    - *'p': [1,2]*
- Regressão Logística (*Logistic Regression*):
    - *‘tol': [0.0001,0.00001,0.000001],*
    - *'C': [1.0,1.5,2.0],*
    - *‘solver':['newton-cg','lbfgs','liblinear','sag','saga']*
- *SVM:*
    - *'C': [1.0,2.0],*
    - *'kernel':['linear', 'poly', 'rbf', 'sigmoid', 'precomputed']*
- Perceptron Multicamadas (MLPClassifier):
    - *'max_iter': [100,200,500],*
    - *‘solver’:['adam','lbfgs','sgd'],*
    - *‘activation’:['identity', 'logistic', 'tanh', 'relu'],*
    - *'tol':[0.0001,0.00001]*

Vale ressaltar que alguns modelos tiveram seus códigos rodando no GridSearchCV, por aproximadamente 48 horas. Entretanto o modelo de SVM não passou pela validação cruzadas, por conta de ficar rodando várias horas e reiniciar o computador.

5 CONCLUSÃO

Neste presente trabalho teve como objetivo aplicar diferentes modelos de *machine learning* e fazer a comparação de seus desempenho utilizando uma base de dados referente a qualidade do ar dos detritos de Beijing.

Posteriormente, após todos os processos de limpeza e análise dados, foi realizado os treinos e testes. Utilizamos vários modelos como Árvore de Decisão (*Decision Tree*), Rede Neural Artifical (*RNA*), Regressão Logística, *KNN*, *SVM* e a Floresta Aleatória (*Random Forest*), no entanto, obtivemos os melhores resultados com os modelos de Floresta Aleatória (*Random Forest*), em todas as configurações de teste esse modelo foi superior aos outros, a métrica mais levada em consideração neste presente trabalho foi a *F1_Score*. Esse métrica faz a média ponderada de outras duas métricas, que são a *precision e recall*, sendo assim uma melhor confiabilidade na medida dos desempenhos

Os modelos utilizados podem ser aplicados a diferentes cenários com as mais diferentes bases de dados possível, tendo em vista que a análise e limpeza de dados inicial é uma das partes mais importantes, prosseguimento do trabalho ficaria similar.

Portanto, pode-se concluir que o modelo de *Random Forest* teve o melhor desempenho dentro todos os modelos testados por conta da sua adaptabilidade a aleatoriedade, pois o modelo de Árvore de Decisão (Decison Tree) tem uma alta chance de Overffing, que é o ajustei fino a base de treino e desempenho ruim na base de teste. Já o modelo de Floreta Aleatória (Random Forest) difere por possuir a aleatoriedade, na escolha de seus atributos para construção das pequenas Árvores de Decisão que o modelo utiliza, após a geração dessas arvores com atributos combinados de forma aleatórias, ele faz uma votação majoritária para escolher qual será a classificação dos dados testado. Dessa forma, o modelo obteve os seguintes resultados organizados por configuração utilizada, grau assertivo e tempo de treinamento: conf_1 - 88% - 00:06:51, conf_2 - 62% - 00:18:09, conf_3 - 44% - 00:30:32, conf_4 - 38% - 00:46:14, conf_5 - 44% - 01:01:38, conf_6 - 42% - 01:13:07.

5.1DESENVOLVIMENTO FUTUROS

Como planejamento para projetos futuros, é a criação de uma nova coluna no *data frame* atual, que iria representar a qualidade do ar, tendo alguma informações pré-configuradas, com as variáveis utilizadas no presente trabalho, como PM_{10}, $PM_{2.5}$, SO_2, NO_2, O_3, CO, até determinada quantidade de volume de cada substância na atmosfera qualidade Boa do ar, caso a quantidade esteja acima do valor, a qualidade é Ruim.

BIBLIOGRAFIA.

- [1] Downey, A. B. Think stats exploratory data analysis in python. Version 2.1.0. – Ultimo acesso em: 20/05/2021.
- [2] McKinney, W. Python para análise de dados - tratamento de dados pandas, numpy e ipython. Novatec Editora; 1ª edição (maio 2019). - Acesso em: 20/05/2021.
- [3] Peng, R. D, Matsui, E. The Art of Data ScienceA - Guide for Anyone Who Works with Data.
- [4] A Programmer's Guide to Data Mining: The Ancient Art of the Numerati
- [5] VanderPlas, J. Python Data Science Handbook: Essential tools for working with data. O'Reilly Media; 1ª edição (21 novembro 2016)
- [6] Goodfellow, I; Bengio, Y; Courvelle, A. Deep Learning. 2017 MIT.
- [7] Grus, Joel. Data Science do Zero. O'Reilly Media; 1ª edição (21 novembro 2016)
- [8] https://archive.ics.uci.edu/ml/datasets/Beijing+Multi-Site+Air-Quality+Data (Uci Dataset - Base de Dados) - Último acesso - 05/09/2021.
- [9] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5627385/ - Último acesso: 31/07/2021
- [10] https://brasil.elpais.com/brasil/2015/12/07/internacional/1449490356_143778.html - Último acesso - 01/08/2021
- [11] https://www1.folha.uol.com.br/mercado/2020/12/retomada-puxada-pela-industria-explica-maior-poluicao-em-pequim-diz-relatorio.shtml - Último acesso - 01/08/2021
- [12] https://www.comciencia.br/uma-breve-trajetoria-da-questao-ambiental-recente-na-china/#_ftn8 - Último acesso - 01/08/2021
- [13] http://climacom.mudancasclimaticas.net.br/discussoes-sobre-a-questao-ambiental-na-china-impactos-e-perspectivas/ - Último acesso - 01/08/2021
- [14] https://pandas.pydata.org - Último acesso – 05/09/2021.
- [16] https://numpy.org - Último acesso – 05/09/2021.
- [17] https://seaborn.pydata.org- Último acesso – 05/11/2021.
- [18] https://scikit-learn.org/stable/- Último acesso – 05/11/2021.
- [19] https://matplotlib.org- Último acesso – 05/11/2021.