Sobre o uso de inteligência artificial e machine learning para predição de marés e propor o melhor arranjo das etapas do circuito nacional de surf

Um estudo de caso

João Pedro Kalil Coelho Lyra

Relatório de Projeto Final de Graduação

Departamento de Informática
Centro Técnico Científico – CTC
Curso de Graduação em Ciência da Computação

Rio de Janeiro
06 de 2021

João Pedro Kalil Coelho Lyra

Uso de inteligência artificial e machine learning para predição de marés e propor o melhor arranjo das etapas do circuito nacional de surf

Relatório de Projeto Final II, apresentado ao programa de **Ciência da Computação** da PUC- Rio como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Edward Hermann Haeusler

Rio de Janeiro
06 de 2021

Resumo

Kalil Coelho Lyra, João Pedro. Hermann, Edward. Uso de inteligência artificial e machine learning para predição de marés e propor o melhor arranjo das etapas do circuito nacional de surf. Rio de Janeiro, 2021, XXp. Relatório de Projeto Final 2. Pontifícia Universidade Católica do Rio de Janeiro.

Este projeto tem a finalidade de utilizar dados de maregráficos e metereológicos para que seja possível criar modelos de predição com o intuito de prever e classificar na tentativa de estender a atual janela de 15 dias de previsão de ondas.

Palavras Chave

Surf, Machine Learning, Redes Neurais, Inteligência Artificial, Onda, Maré

Sumário

1 - Introdução

Atualmente, os circuitos de surf que temos espalhados pelo mundo, inclusive o nacional, seguem sempre a mesma sequência de etapas independentemente das condições climáticas ou maregráficas, ou seja, esteja o mar bom ou não, aquela etapa ocorrerá. Quando as condições estão boas, o campeonato ocorre normalmente, porém, quando o mar não está propício para a prática do esporte o campeonato entra no estado que chamamos de “on hold” onde os participantes esperam algumas horas ou dias até que ele volte a estar em condições para uma boa prática do surf.

Tendo isso em vista, os órgãos organizadores dos campeonatos geralmente já escolhem os locais das etapas de acordo com as épocas de melhores ondas em cada local, por exemplo, aqui no Brasil, a melhor época é no inverno, então os organizadores como a WSL (órgão organizador do campeonato mundial) colocam a etapa brasileira no período de maio/junho. O grande problema de ter etapas com datas já definidas sempre na mesma ordem e no mesmo período do ano é que pode acontecer de ter mais dias “on hold” do que dias de competição de fato, ou competição forçada em um mar ruim para praticar o surf. Isso tudo acaba prejudicando a experiência da competição tanto para quem está assistindo quanto para quem está competindo, é pior para o evento pois a organizadora acaba gastando mais dinheiro com a estrutura pelo fato dela acabar ficando mais dias do que o previsto, logo tem que pagar os funcionários locais por mais dias de trabalho, tem que estender o contrato de aluguel dos equipamentos da estrutura física, da equipe de segurança, dos geradores de energia entre outras coisas que geram muitos gastos a mais.

O trabalho tem como objetivo principal fazer um estudo de inteligência artificial e aprendizado de máquina na presença de dados oceanográficos de baixa qualidade e em baixa quantidade visando tentar criar um modelo de classificação onde, baseado em uma localidade geográfica com suas condições temporais ideais, como vento e direção do vento, e condições maregráficas ideia como ondulação, direção da ondulação e força da ondulação, visando dizer se uma condição prevista será ideal para poder surfar. Tendo esse modelo para um período básico de previsão de 15 dias, o objetivo é ir aumentando essa janela para chegarmos a um modelo que seja possível prever e classificar pelo menos um mês inteiro. O fato de ter a janela de um mês inteiro previsto é que, apesar de não ser o ideal, uma organização de surf poderia fazer seu calendário dinamicamente no decorrer do ano, ao invés de montar o mesmo no início do ano e correr o risco de em alguma etapa ter más condições de surf.

Para o desenvolvimento foi utilizado, no início, um notebook com windows 10 e posteriormente um macbook Pro com chipset M1 da Apple com o MacOS BigSur como sistema operacional. Junto a isso, foi utilizado a IDE Spyder e o Jupyter Notebooks para a escrita do código, limpeza, tratamento e análise dos dados. Para rodar os modelos criados no projeto foi utilizado o Google Colaboratory, um ambiente de desenvolvimento igual ao do Jupyter porém rodado em cloud, utilizando-se do poder computacional das máquinas dos servidores da Google. As bibliotecas pandas, Scikit-Learn foram as principais para as análises e modelagens. Além disso, foram utilizadas as bibliotecas Pandas e Numpy para a melhor manipulação dos dados, e as bibliotecas Matplotlib e Seaborn para uma melhor experiência na visualização dos dados.

Esse projeto encaixa-se muito bem como projeto final pois põe em prática vários assuntos vistos dentro das salas de aula da PUC-Rio, como: Inteligência Artificial, Aprendizado de máquina, programação e uso de linguagem de programação, ciência de dados e modelagem de dados. O projeto serviu e está servindo como uma forma de aplicar na prática os assuntos que foram muito bem estudados na teoria porém com pouco tempo para executar.

2 – Situação Atual

Atualmente, na internet existem sites que mostram tábuas de marés, previsão de ondas, tamanho de ondas, vento entre outros dados que geralmente são mostrados quando surfistas olham para verificar se no dia, próximo àquele, terá boas ondas. O funcionamento mais comum dentro dos sites com software de previsão de ondas é: pegar os dados meteorológicos que são captados através de satélites, onde há a capacidade de análise de dados global, ou de estações meteorológicas, que fazem as análises locais de onde elas estão posicionadas, para então serem inseridos em um modelo matemático oceanográfico que efetuará os devidos cálculos para que as previsão das condições de ondas seja feita. O modelo mais usado atualmente, inclusive pelo site parceiro do projeto, o Surfguru, é o WAVEWATCH III. Esse é um modelo de onda de terceira geração desenvolvido na NOAA/NCEP (National Center for Environmental Prediction) e adicional do modelo WAVEWATCH e WAVEWATCH II. No entanto, difere de seus predecessores em muitos pontos importantes, como as equações governantes, a estrutura do modelo, os métodos numéricos e as parametrizações físicas. Além disso, após uma de suas versões do modelo, o WAVEWATCH III está evoluindo de um modelo de onda para uma

estrutura de modelagem de onda, o que permite fácil desenvolvimento de abordagens físicas e numéricas adicionais para modelagem de onda.

Juntamente a isso, o NOAA Global Forecast System, um modelo de previsão do tempo do NCEP que gera dados para dezenas de variáveis atmosféricas, incluindo temperaturas, ventos, precipitação, umidade do solo e concentração de ozônio atmosférico, acoplando quatro modelos separados (atmosfera, modelo oceânico, modelo terra / solo e gelo marinho) que funcionam juntos para representar com precisão as condições climáticas, é utilizado para pegar os dados de previsão atmosférica utilizados também nos cálculos que geram as previsões do site. E por fim, mas não menos importante, as tabelas de maré fornecidas pela Marinha do Brasil também são incorporadas para o resultado final. Contudo, toda essa junção de dados de vários lugares diferentes acabam em um output de somente 15 dias de previsão de ondas, que, apesar de ser o mais comum dentre esses softwares, são pouquíssimos dias de resultado.

Voltando as atenções para os dados tivemos, em todos os conjuntos que manipulamos, todos eles muito bem rotulados, apesar de, nos dados exportados pela marinha do Brasil e o dados fornecidos pela API do INMET, ter informações faltando sendo mostrado, nas inúmeras linhas, valores ***null***. Os mesmos dados não se mostraram bem organizados, apesar de bem rotulados. Notamos que os dados referentes a meteorologia do INMET são muito parecidos com os da Marinha do Brasil, mostrando tamanha semelhança principalmente no ponto onde os valores ***null*** começam e terminam. Já o Surfguru se mostrou diferente. Ele, por ter fornecido dados processados ao invés de brutos, tem um nível de consistência e qualidade muito maior que as outras fontes além de estarem muito melhor rotulados e organizados. Como o principal pico de surf estudado foi o de Saquarema, tivemos a oportunidade de validar se os dados eram precisos já que um de nós reside na cidade.

3 – Objetivo do trabalho

O principal objetivo do projeto é construir um software para prever condições de ondas em locais específicos, para tentar estender a janela de previsão de ondas de 15 para 30 dias e assim sucessivamente, até chegar em um ponto que não seja mais possível estender a mesma. Para isso será necessário fazer um estudo mais aprofundado de como o mar se comporta perante as condições climáticas, quais as

influências diretas e indiretas dos parâmetros climáticos nos oceanos, como são feitas as correlações dos parâmetros climáticos e maregráficos e como funcionam os modelos matemáticos por trás dos softwares que existem hoje no mercado.

Com um escopo relativamente pequeno, o projeto contém 2 programas principais. O primeiro deles tem como objetivo gerar um arquivo que contenha os dados de medições de cada dia dos anos desde 2008 até hoje feitas pelo software do site do Surfguru. Para obter essas informações foi necessária a ajuda da biblioteca Selenium, que ajudou na navegação e extração de dados de websites.

Um segundo programa precisa, como input inicial, do arquivo gerado pelo primeiro, citado no parágrafo acima. Com os dados brutos carregados, o programa trabalha neles para gerar novos campos que irão ajudar na composição desse mesmo grupo de dados para então serem utilizados como base para o desenvolvimento de um modelo de regressão em primeira instância mostrando quais features tem mais correlação e quais são as mais importantes para o modelo. Seguindo da utilização do resultado em um modelo de rede neural recorrente chamado Long Short Term Memory, ou LSTM, que será citada mais detalhadamente como funciona no tópico 6.3.1 abaixo.

4 – Atividades realizadas

Para o desenvolvimento do projeto foi necessário, conhecimentos prévios de inteligência artificial, machine learning e seus vários métodos de aplicação e desenvolvimento. Assim como conhecimento sobre oceanografia, como formação de ondas, influência do tempo na formação das mesmas. Algum conhecimento de meteorologia e, também, conhecimento da linguagem de programação Python. Com isso é possível separar o conjunto de atividades realizadas dentro do cronograma em 3 etapas.

A organização e desenvolvimento do projeto foi baseada na metodologia do Scrum, porém não seguido à risca, uma vez que esse método é para equipes com 4 ou mais pessoas. A ideia de utilizar algumas práticas da metodologia ágil surgiu pelo fato de que o escopo inicial era simples devido à falta de conhecimento geral dos assuntos que envolviam o projeto e que, ao longo do estudo, aprendizado e desenvolvimento, o escopo foi se modificando, seja aumentando ou diminuindo.

4.1 – Primeira Etapa

Como foi dito acima, foram necessários alguns estudos para que nós tivéssemos

o mínimo de conhecimento para entender o que queríamos fazer e poder começar a desenvolver o software. Começamos pelo estudo de inteligência artificial, pois como já era algo que já havíamos estudado superficialmente, decidimos que seria o primeiro no qual gostaríamos de nos aprofundar. Buscamos então por cursos online e material suplementar na web para incrementar nosso conhecimento e fundamentar melhor o projeto. Os assuntos abordados variam de “**data processing**” que é o básico para que tenhamos dados consistentes para trabalhar, até assuntos mais complexos como, por exemplo, “**Natural Language Processing**” e "**Long Short Term Memory (LSTM)**". Como parte do nosso estudo envolve fazer uma climatologia de ondas, inicialmente focamos bastante na parte de classificação do curso, onde é ensinado alguns algoritmos de classificação como **K-Nearest Neighbors (K-NN), Kernel SVM, Support Vector Machine (SVM), Logistic Regression, Decision Tree Classification, Naive Bayes, Random Forest Classification**, para que então através da nossa análise e junção de dados, fosse possível dizer se as condições que teríamos em um certo pico de surf, dado as condições de dados brutos, seriam favoráveis para a prática do esporte e também tivemos esforços voltados para o estudo de complexidade amostral para que fosse possível a compreensão dos resultados obtidos frente aos modelos de machine learning utilizados.

4.2 – Segunda Etapa

Passando para o próximo tópico estudado, seguimos então para o estudo de oceanografia e meteorologia. Começamos então buscando informações sobre dados com a comunidade de oceanografia, para ter uma base de onde começar. Seguimos, primeiramente, por como as marés são calculadas e previstas, seguido do estudo da influência do clima na formação de ondas juntamente com o estudo de fundos do oceano e como a forma deles influencia na quebra de uma onda no litoral. Para aprofundarmos ainda mais os estudos, buscamos por dados maregráficos e climatológicos da marinha do Brasil. Através de trocas de e-mails constantes, foi possível acessar os dados de forma padronizada e organizada. Contudo, após uma série de análises e filtros aplicados aos dados de nossa posse, descobrimos que os dados eram intermitentes tendo, às vezes, mais medições que o normal e vezes que os sensores das estações maregráficas falhavam, acarretando na não medição e menor quantidade de dados em um dia. Tentamos ver o que conseguiríamos aproveitar desses dados para que contribuíssem com o desenvolvimento do projeto, mas acabou que optamos por descartá-los e seguir

para uma nova fonte mais consistente. Descobrimos então o INMET.

O INMET, Instituto Nacional de Meteorologia, diferentemente da marinha e mais moderno, possui uma API para o acesso aos seus dados bem documentada, acarretando então na necessidade de termos que estudá-la para o devido uso. Após o estudo e os devidos testes da mesma através da ferramenta Postman em paralelo com o Jupyter Notebooks, que foi utilizado para fazer os scripts de limpeza e de tratamento de dados, concluímos também, que não daria para ser a nossa principal fonte de dados para o projeto, mas que não descartamos a possibilidade de usá-la como complemento.

Finalizamos essa etapa com o fechamento de uma parceria com a empresa Surfguru. Apresentamos a ideia para eles, que por sua vez gostaram muito e decidiram apostar no projeto. Com isso iniciamos uma etapa de obtenção dos dados da base deles, que apesar de não ter uma API que retornasse JSON, tinha um retorno formatado, facilitando para a extração dos dados, mostrados no browser, através de um web crawler que desenvolvemos em Python utilizando a lib Selenium como comentamos na seção anterior. E posteriormente a seção de formatação dos dados para um padrão legível para os nossos futuros modelos de machine learning.

4.3 – Terceira Etapa

Como terceira e última etapa, seguimos para o desenvolvimento dos nossos modelos de machine learning, que aprendemos nos cursos da PUC e nas fontes complementares que citamos na seção 4.1, utilizando os dados que obtivemos na etapa anterior. Escolhemos a dedo alguns modelos para que fizéssemos os devidos testes e analisássemos como cada um se comportaria frente aos nossos conjuntos de dados. Utilizamos modelos como Random Forest, Naive Bayes, Logistic Regression e LSTM para efetuar as classificações, já para as predições optamos pelos modelos de Random Forest e Polinomial Regression.

4.4 – Cronograma

O primeiro cronograma feito contemplava, no primeiro semestre do ano, o estudo aprofundado sobre os devidos temas de machine learning e inteligência artificial, com o estudo de oceanografia e as bibliotecas de python que seriam utilizadas como as principais ferramentas. Em paralelo a busca por dados, que no fim, acabou sendo mais longa do que o planejado inicialmente. Algumas tentativas de criação de modelos simples

de classificação e predição, com os dados que foram obtidos através da Marinha do Brasil e do INMET, foram feitas, porém sem muito sucesso, uma vez que os mesmos eram inconsistentes e irregulares.

Para o segundo semestre, acabamos adaptando o cronograma para a realidade encontrada, com todos os problemas e atrasos que tivemos com a obtenção de dados. O cronograma acabou tendo mais algumas etapas que envolveriam a base do Surfguru, sendo elas, a extração e tratamento dos dados dessa base e o estudo e criação dos modelos de classificação e predição em cima dos mesmos. Acabamos implementando alguns modelos que não estavam planejados inicialmente e nos empenhamos bastante em um modelo de deep learning, que apesar de termos poucos dados, se mostrou com um bom percentual de acurácia. Abaixo temos a legenda e a tabela do primeiro cronograma que fizemos, seguido com o cronograma planejado do segundo semestre e então o cronograma executado no segundo semestre:

- Cronograma inicial planejado e executado no primeiro semestre

	Meses											
Atividades	Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	Nov	Dez
Estudo de IA, Machine Learning, oceanografia e libs de Python	✔	✔	✔	✔	✔	✔	O					
Busca por dados	✔	✔										
Limpeza / tratamento dos dados		✖										
Seleção de Features			✖	✖	✖	✖						
Criação de modelos simples				✖	✖							
Proposta de PF 1				✔								
Relatório de PF1						✔	✔					
Criação do modelo complexo							✖	O	O			
Estudo dos resultados e devidos consertos								O	O			
Testes com anos anteriores									O	O		
Relatório PF 2										O	O	
	Legenda											
	✖ → não feito ✔ → feito O → planejado											

Figura 1 - Cronograma primeiro semestre

- Cronograma do segundo semestre planejado

	Meses											
Atividades	Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	Nov	Dez
Tratamento dos dados do Surfguru						O	O					
Criação de modelos de classificação simples							O	O				
Estudo de features para aprimoramento dos modelos de classificação								O	O			
Criação do modelo complexo para auto-aprendizagem								O	O			
Estudo dos resultados e devidos consertos									O	O		
Testes com anos anteriores										O		
Relatório PF 2										O	O	

Figura 2 - Cronograma planejado segundo semestre

- Cronograma do segundo semestre executado

	Meses											
Atividades	Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	Nov	Dez
Tratamento dos dados do Surfguru						✔	✔					
Criação de modelos de classificação simples							✔	✔				
Estudo de features para aprimoramento dos modelos de classificação								✖	✔			
Criação do modelo complexo para auto-aprendizagem								✖	✖	✔		
Estudo dos resultados e devidos consertos									✖	✔		
Testes com anos anteriores										✔		
Relatório PF 2										✖	✔	
	Legenda											
	✖ → não feito ✔ → feito O → planejado											

Figura 3 - Cronograma executado segundo semestre

5 – Feature Engineering

Praticamente todos os algoritmos de machine learning possuem entradas e saídas. As entradas são formadas por colunas de dados estruturados, onde cada coluna recebe o nome de feature, também conhecido como variáveis independentes ou atributos. É muito comum em projetos que envolvem machine learning não termos 100% definido o que usaremos como features para os nossos futuros modelos. Então, para essa etapa dos projetos, existe o que chamamos de "Feature Engineering", que traduzido fica "Engenharia de Recursos". Como definição, engenharia de recursos é o processo de usar o conhecimento de um domínio para extrair recursos de dados brutos. Atos como brainstorm, criação e teste de features são parte de um fluxo comum de Feature Engineering. É a partir desse processo que identificamos dentro do nosso grupo de dados, por exemplo, qual dos atributos é o mais relevante deles, ou seja, qual que teve o maior impacto para o nosso modelo na hora dele tomar as decisões, seja nas suas classificações ou predições.

Dado o contexto, podemos citar as features engineering que mais utilizamos dentro do nosso desenvolvimento. São elas: feature scaling e média.

5.1 – Feature Scaling

O feature scaling é um método usado para normalizar o intervalo de atributos independentes ou recursos de dados. É, também, conhecido como normalização de dados e é, geralmente, realizado durante a etapa de pré-processamento de dados. É muito comum quando temos uma faixa de valores nos dados brutos que varia amplamente, fazendo com que alguns algoritmos de aprendizado de máquina não funcionem corretamente sem normalização. Por exemplo, muitos classificadores calculam a distância entre dois pontos pela distância euclidiana. Se um dos recursos tiver uma ampla faixa de valores, a distância será controlada por esse recurso específico. Portanto, o intervalo de todos os recursos deve ser normalizado para que cada recurso contribua aproximadamente proporcionalmente para a distância final.

Existem algumas formas de se fazer feature scaling. O método que utilizamos em nosso projeto foi o Standard Scaling, feito pela classe Standard Scaler da nossa biblioteca utilizada, Scikit Learn. Esse método é conhecido, também, como

Standardization ou Z-Score Normalization, e funciona determinando a distribuição média e o desvio padrão para cada recurso. Em seguida, subtrai-se a média de cada recurso e em seguida, divide-se os valores (a média já foi subtraída) de cada característica por seu desvio padrão. Abaixo, a descrição matemática do método:

$$x' = \frac{(x - \bar{x})}{\sigma}$$

onde **x** é o vetor da característica original, $\bar{x}$ é a média desse vetor de característica, e **σ** (sigma) é seu desvio padrão.

5.2 – Média

Esse método é mais simples que o citado anteriormente. Ele consiste, basicamente, em pegar a série temporal de medições de um dia, no seu intervalo de horas, e aplicar a média aritmética sobre a mesma. Apesar de simples, teve grande relevância na hora da criação das nossas features e modelos. Creio que a média tenha performado bem devido à relação dos dados entre sim de cada atributo. Por exemplo, o tamanho de onda em um dia não costuma variar muito, a não ser em casos especiais de mudança drástica do tempo como a entrada de uma frente fria. Então fazer uma média do tamanho de onda do dia é mais simples e assertivo do que tentar usar outro método de feature engineering.

6 – Machine Learning

Quando o assunto é machine learning temos dois tipos de técnicas para treinar nossos modelos, são eles: aprendizado supervisionado, que treina um modelo em dados de entrada e saída conhecidos para que ele possa prever saídas futuras, e aprendizado não supervisionado, que encontra padrões ocultos ou estruturas intrínsecas nos dados de entrada.

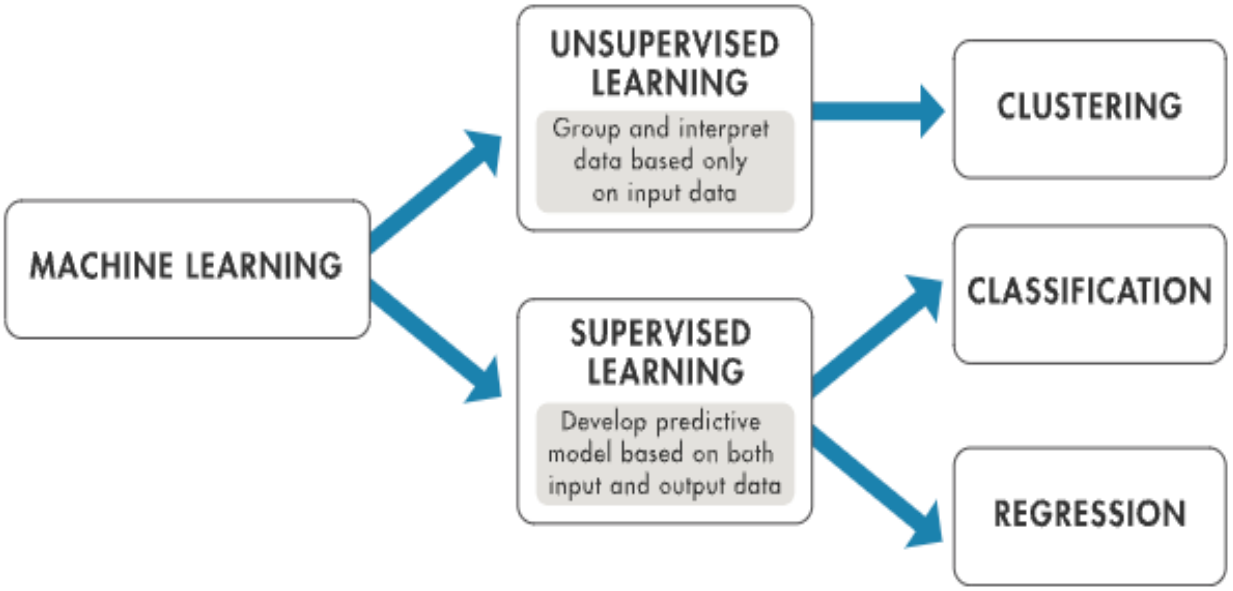

Figura 4 - Ramos do machine learning [17]

Como em nosso projeto temos que fazer previsões e classificações, tivemos que ir pelo caminho do aprendizado supervisionado. O aprendizado de máquina supervisionado constrói um modelo que faz previsões com base em evidências na presença de incerteza. Um algoritmo de aprendizado supervisionado pega um conjunto conhecido de dados de entrada e respostas conhecidas aos dados (saída), também conhecidas por rótulos, e treina um modelo para gerar previsões razoáveis para a resposta a novos dados. Como mostrado na imagem acima, dentro da área de algoritmos supervisionados, nós possuímos dois tipos de algoritmos dos quais implementamos no nosso projeto. São eles: algoritmos de regressão e de classificação e podemos ver alguns exemplos na imagem seguinte.

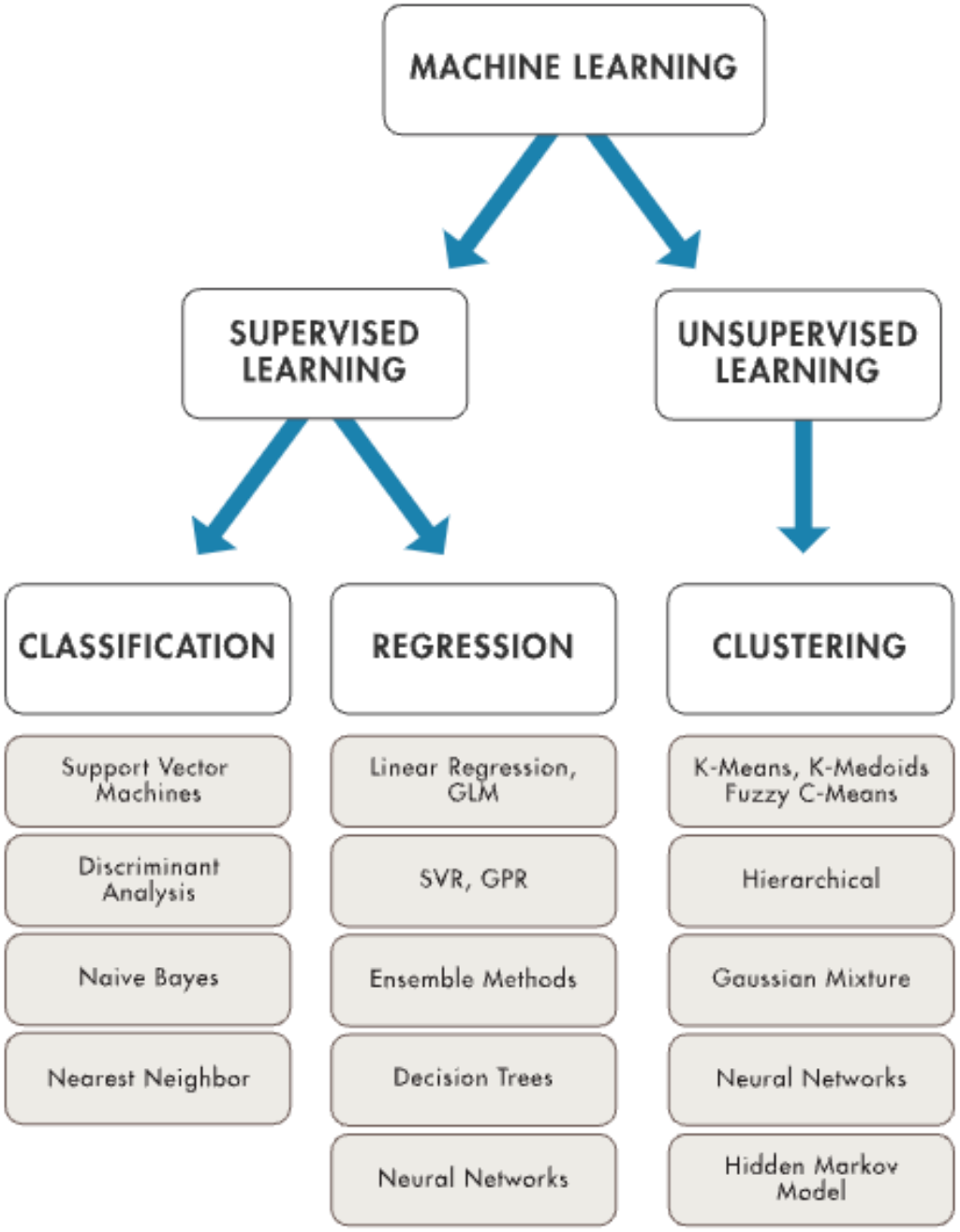

Figura 5 - Tipos de modelos de acordo com tipo de aprendizado de máquina [17]

6.1 – Algoritmos de Regressão

Os algoritmos de regressão preveem respostas contínuas como: mudanças na temperatura, tamanho de ondas, condições climáticas, etc. Usamos técnicas de regressão quando estamos trabalhando com um intervalo de dados ou se a natureza de

sua resposta for um número real, como temperatura ou o tempo até a falha de um equipamento. Dentro dos modelos de regressão temos os conceitos de variáveis dependentes e independentes, que são as que nos ajudam a detectar os padrões e aprender com os mesmos e as que nós estamos querendo prever, respectivamente.

Não falamos sobre regressão linear de forma aprofundada aqui pois não entrou no âmbito do projeto, porém, precisamos entender como o modelo mais simples de regressão funciona para que seja possível entender dos modelos mais complexos que iremos falar aqui.

6.1.1 – Linear Regression

O objetivo de um modelo de regressão linear é encontrar uma relação entre uma ou mais variáveis independentes e uma variável dependente. Um modelo de regressão linear pode ser descrito da forma:

$$Y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n$$

Onde **Y** é o valor que estamos querendo prever, θ_0 é o termo bias, $\theta_1,\ldots,\theta$ são os parâmetros do modelo e $x_1, x_2,\ldots,x$ são os valores das features. O problema de um modelo de regressão linear é devido a alta correlação das variáveis independentes com as dependentes, fazendo com que não tenhamos boas previsões quando estamos lidando com dados mais complexos e dispersos. Podemos ver, mais claramente, nas imagens abaixo:

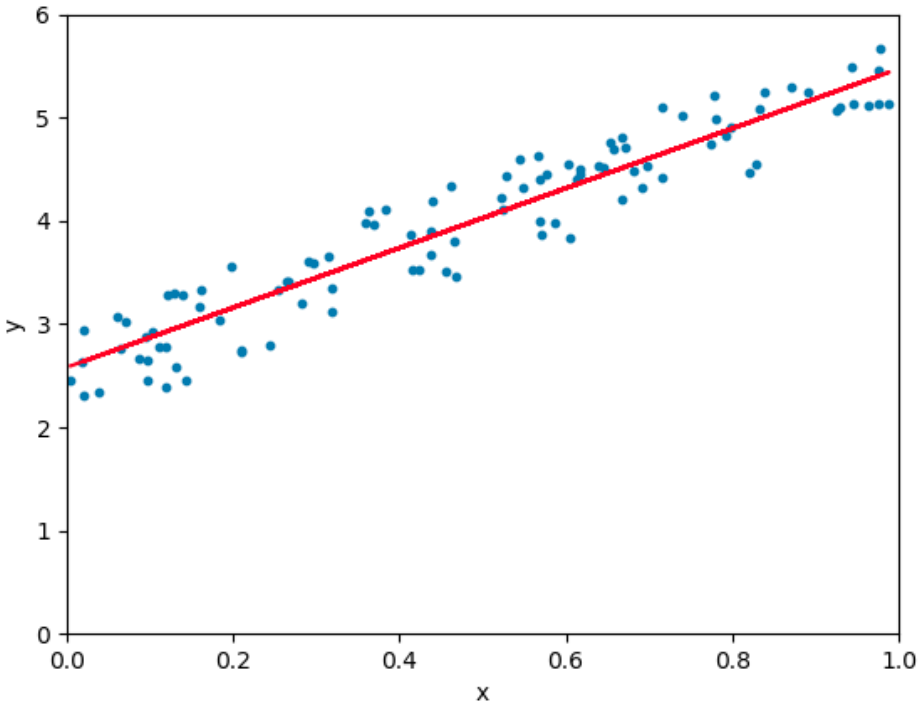

Figura 6 - Comportamento de uma regressão linear [2]

6.1.2 – Polynomial Regression

Esse é um modelo apropriado quando os modelos lineares possuem uma performance abaixo do esperado, indicando que os dados talvez não forneçam um padrão linear. A diferença desse modelo para os modelos lineares é que as variáveis dependentes são modeladas em um grau polinomial N, especificando a sua relevância. Podemos ver como um modelo polinomial pode ser descrito.

$$Y = \theta_0 + \theta_1 x + \theta_2 x^2$$

Conseguimos ver, na imagem abaixo, os dois modelos aplicados em um conjunto de dados, sem correlações lineares e dispersos de forma complexa, e ver como o modelo polinomial se adequa muito melhor ao conjunto.

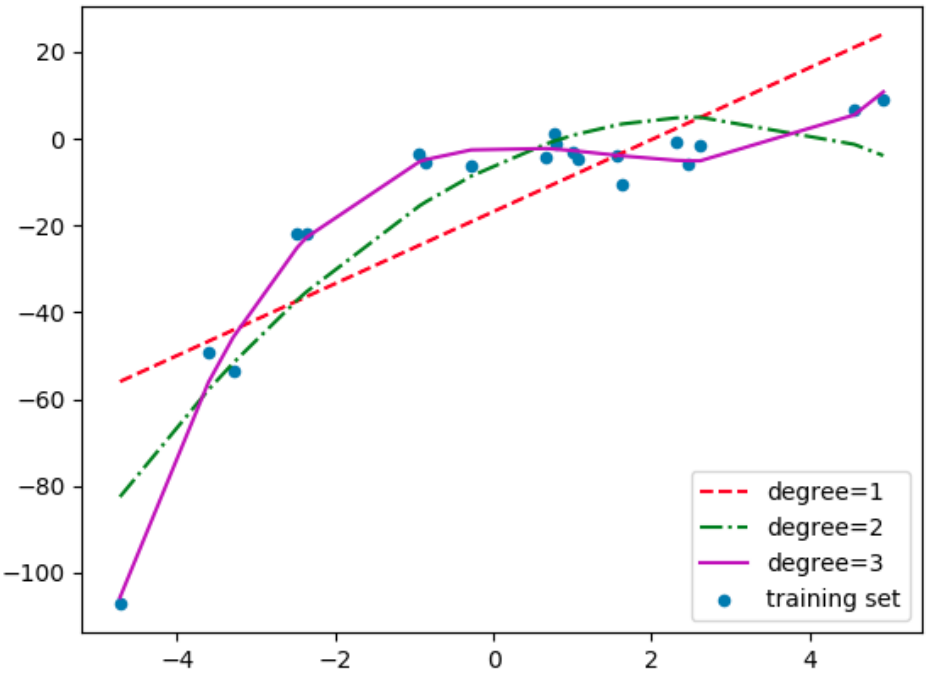

Figura 7 - Comportamento de um regressão polinomial [16]

6.1.3 – Random Forest Regression

Random Forest é um algoritmo que se baseia no método de aprendizagem em conjunto e em muitas árvores de decisão. No Random Forest todos os cálculos são executados em paralelo e não há interação entre as Árvores de Decisão ao construí-las.

A árvore de decisão é um algoritmo de fácil compreensão e interpretação e, portanto, uma única árvore pode não ser suficiente para o modelo aprender os recursos dela. Por outro lado, Random Forest também é um algoritmo baseado em “árvore” que usa os recursos de qualidades de várias Árvores de Decisão para tomar decisões. Portanto, pode ser referido como uma "Floresta" de árvores e, portanto, o nome "Random Forest". O termo 'Random' se deve ao fato de que este algoritmo é uma floresta de Árvores de Decisão criadas aleatoriamente.

Uma floresta aleatória opera construindo várias árvores de decisão durante o tempo de treinamento e produzindo a média das classes como a previsão de todas as árvores. Primeiro ele escolhe k pontos de dados aleatórios do conjunto de treinamento, logo depois, uma árvore de decisão associada a esses k pontos de dados. Escolhe-se o número N de árvores que deseja construir e depois ele repete as etapas 1 e 2. Por fim, para um novo ponto de dados, ele faz com que cada uma de suas árvores N-tree

prevejam o valor de y para o ponto de dados em questão e atribui o novo ponto de dados à média de todos os valores de y previstos.

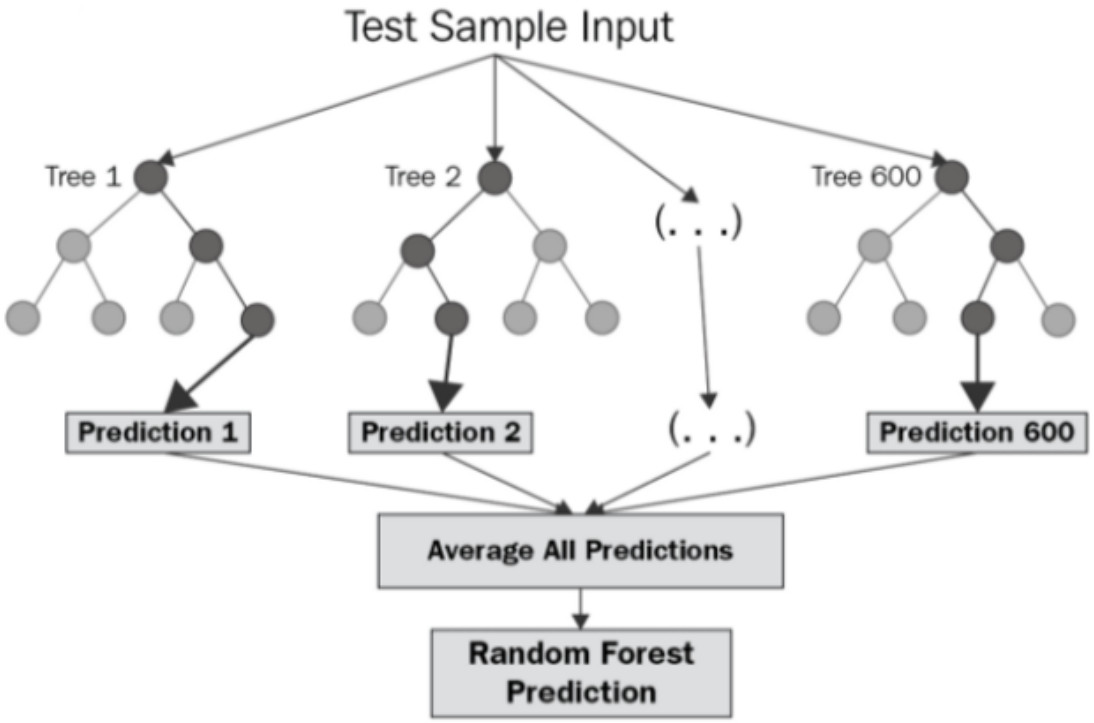

Figura 8 - Funcionamento do Random Forest Regressor [14]

6.2 – Algoritmos de Classificação

Esses algoritmos são utilizados quando um conjunto de dados já classificados é utilizado para construir um modelo capaz de prever a classificação de outros dados futuros e, por isso, também são conhecidos como modelos preditivos. Na prática, algoritmos como este são utilizados para prever um aumento ou uma queda no preço de ações específicas na bolsa de valores, por exemplo. Um detalhe importante ao escolher um algoritmo de aprendizagem supervisionada é ficar atento que alguns deles são para classificação binária (apenas duas classes), como quando recebemos um email e o gmail acusa se ser spam. Nesse caso ele classifica em spam ou não spam. Enquanto outros são generalizáveis para múltiplas classes.

6.2.1 – Random Forest Classifier

O Random Forest Classifier utiliza o conceito de ensemble learning que consiste na ideia de combinar diversos modelos de predição mais simples (**weak learner**), treiná-los para uma mesma tarefa, e produzir a partir desses um modelo agrupado mais complexo (**strong learner**) que é a soma de suas partes. No caso do Random Forest Classifier, um algoritmo de Decision Tree será executado múltiplas vezes.

No algoritmo de Decision Tree, conseguimos alcançar a definição da categoria de uma nova instância a partir de diversas perguntas binárias relacionadas aos seus atributos, resultando em uma árvore.

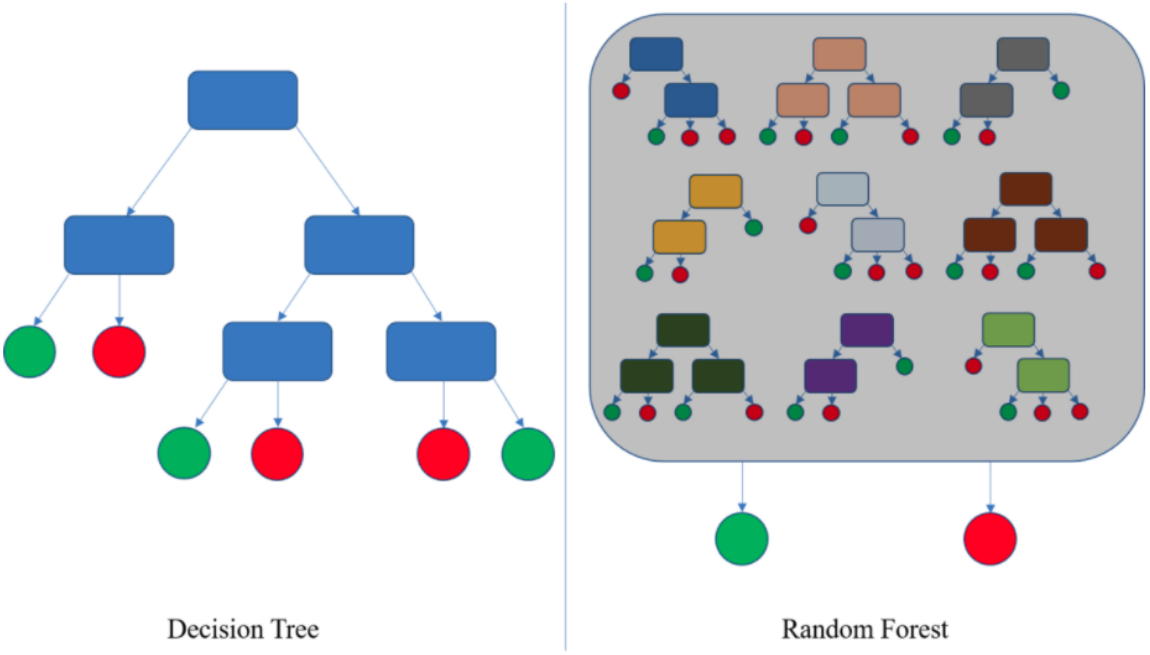

Figura 9 - Funcionamento do Random Forest Classifier [15]

O Random Forest Classifier tem a interessante funcionalidade de correlacionar as features do modelo e gerar um histograma mostrando qual a importância de cada feature no resultado final do modelo. Mostramos isso na figura abaixo.

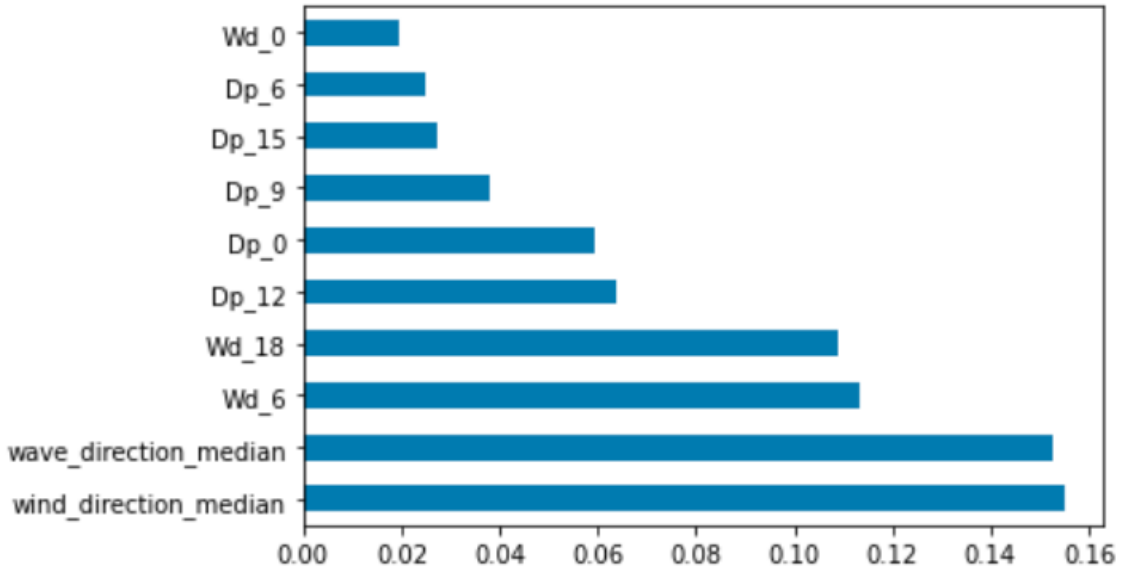

Figura 10 - Features mais relevantes para o Random Forest Classifier

6.2.2 – Naive Bayes

Um classificador Naive Bayes assume que a presença (ou ausência) de uma característica particular de uma classe não está relacionado à presença (ou ausência) de qualquer outro recurso. O exemplo mais comum para explicar o algoritmo é: Pense em uma fruta. Ela pode ser considerada uma maçã se é vermelha, redonda e tem cerca de 4 "de diâmetro. Mesmo que essas características dependam umas das outras ou da existência uma da outra características, um classificador de Bayes considera todas essas propriedades para contribuir de forma independente para a probabilidade de que esta fruta é uma maçã. Uma vantagem do algoritmo de Bayes é que ele requer apenas um pequeno número de dados de treinamento para estimar os parâmetros necessários para a classificação.

A fórmula descrita por Bayes é dada pela imagem abaixo, onde é previsto a probabilidade do conjunto de features X ser do tipo Y. Para chegar nessa solução, multiplica-se a probabilidade de um conjunto de dados do tipo Y pela probabilidade de uma instância de Y ser semelhante à X, dividindo pela probabilidade do conjunto de dados ser semelhante à X.

$$P(y|X) = \frac{P(X|y)\,P(y)}{P(X)}$$

$$P(y|x_1, \ldots, x_n) = \frac{P(x_1|y)\;P(x_2|y) \ldots P(x_n|y)\,P(y)}{P(x_1)\,P(x_2) \ldots P(x_n)}$$

6.2.3 – Support Vector Machines (SVM)

Geralmente, Support Vector Machines é considerado uma abordagem de classificação, mas pode ser empregado em ambos os tipos de problemas de classificação e regressão. Ele pode lidar facilmente com várias variáveis contínuas e categóricas. SVM constrói um hiperplano no espaço multidimensional para separar diferentes classes. O SVM gera o hiperplano ideal de maneira iterativa, que é usado para minimizar um erro. A ideia central do SVM é encontrar um hiperplano marginal máximo (MMH) que melhor divide o conjunto de dados em classes.

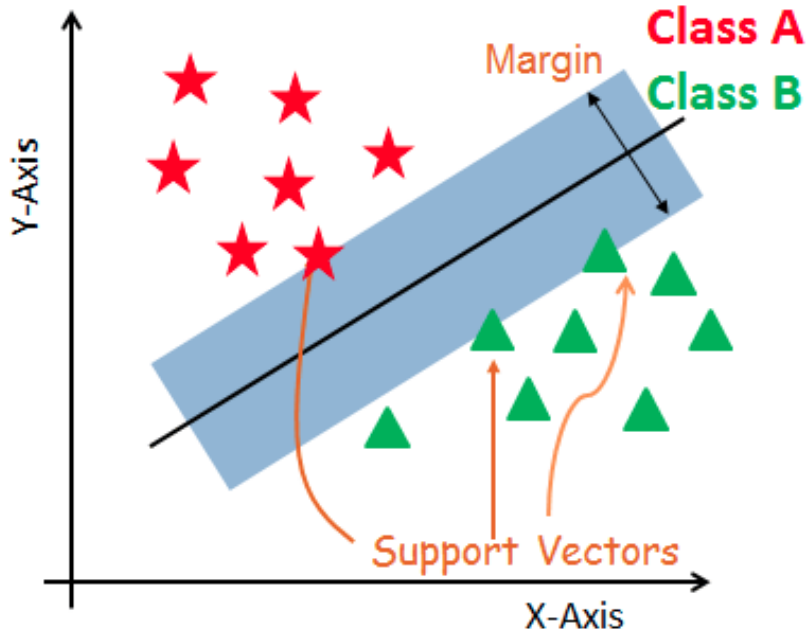

Figura 11 - Support Vector Machine [6]

Os vetores de suporte são os pontos de dados mais próximos do hiperplano. Esses pontos definirão melhor a linha de separação calculando as margens. Esses pontos são mais relevantes para a construção do classificador, além dos vetores, temos o hiperplano que é um plano de decisão que separa um conjunto de objetos com diferentes membros de classe e para finalizar o classificador, temos as margens. Uma margem é uma lacuna entre as duas linhas nos pontos de classe mais próximos. Isso é calculado como a distância perpendicular da linha aos vetores de suporte ou pontos mais próximos. Se a margem for maior entre as classes, então é considerada uma margem boa, uma margem menor é uma margem ruim.

6.3 – Redes Neurais

O objetivo original das redes neurais era criar um sistema computacional capaz de resolver problemas como um cérebro humano. No entanto, com o passar do tempo, os pesquisadores mudaram o foco e passaram a usar redes neurais para resolver tarefas específicas, desviando-se de uma abordagem estritamente biológica. Desde então, as redes neurais são utilizadas nas mais variadas tarefas, incluindo visão computacional, reconhecimento de fala, tradução de máquina, filtragem de redes sociais, jogos de tabuleiro ou vídeo-game e diagnósticos médicos**.**

6.3.1 – Redes Neurais Long Short Term Memory (LSTM)

Como dito acima, a ideia inicial das redes neurais era se comportar como o cérebro humano. Por mais que ao longo do tempo, os pesquisadores tenham se desviado da ideia primária, ainda sim foram desenvolvidas redes que se encaixavam, de certa forma, com o comportamento do nosso cérebro, sendo esse, a capacidade de utilizar memórias recentes para que seja possível utilizá-las junto ao contexto atual tornando compreensível a ação que está ou irá ser tomada. Analogamente a isso, o ato de ler esse texto por nós só é possível pois entendemos cada palavra com base em nossa compreensão das palavras anteriores. Nós não jogamos tudo que lemos ou aprendemos fora e começamos a pensar de novo. Podemos dizer que os nossos pensamentos e conhecimentos têm persistência. Redes neurais tradicionais não são capazes de tal ato.

Entramos então no conceito de redes neurais recorrentes. São redes com loops,

permitindo, de certa forma, a persistência da informação. No diagrama abaixo, um pedaço de rede neural olha para uma entrada ***xt*** gera um valor ***ht***, contudo, ela não para somente nisso. Um loop permite que as informações sejam passadas de uma etapa da rede para a próxima.

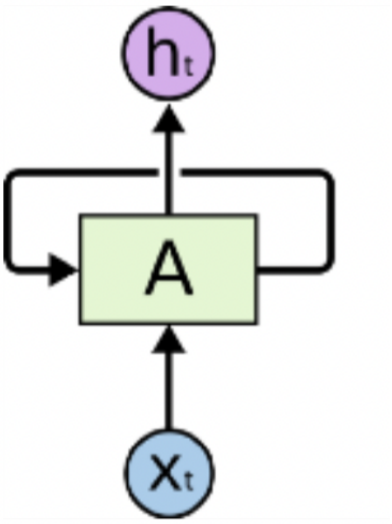

Figura 12 - Loop Rede Neural Recorrente [11]

Uma rede neural recorrente pode ser imaginada como múltiplas cópias da mesma rede, cada uma passando uma mensagem a um sucessor. A figura abaixo ilustra bem o desenrolar da rede.

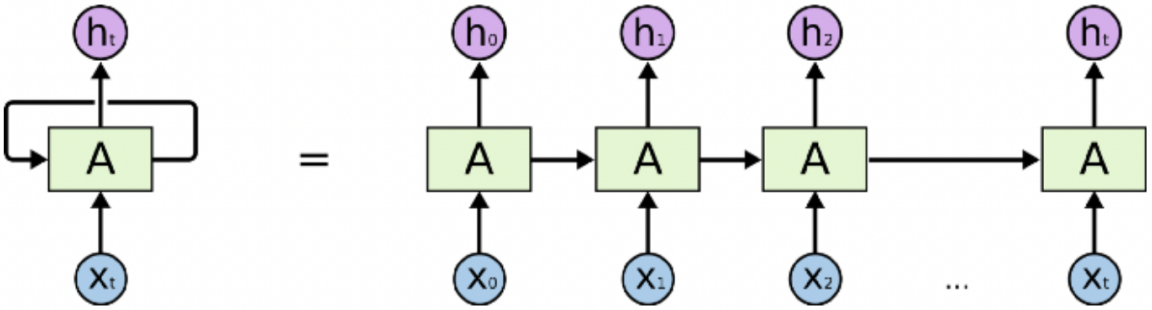

Figura 13 - Loop Rede Neural Recorrente destrinchado [11]

O caso mais conhecido de sucesso nas redes neurais recorrentes são as Long Short Term Memory ou LSTMs. Elas foram introduzidas por Hochreiter & Schmidhuber em 1997, sendo desenvolvida e evoluída por muitos outros autores ao longo dos anos. Utilizadas em diversas áreas, porém, muito comum em previsões de preços de ações no

mercado financeiro, gerações de textos, chatbots, entre outras tantas. A LSTM é uma arquitetura de rede neural recorrente que “*lembra”* valores em intervalos arbitrários fazendo dela uma rede melhor que as outras RNNs mais simples. A LSTM é bem adequada para classificar, processar e prever séries temporais com intervalos de tempo de duração desconhecida.

É importante destacar como é a arquitetura de uma LSTM. Ela é formada por uma estrutura em cadeia que contém quatro redes neurais e diferentes blocos de memória chamados células.

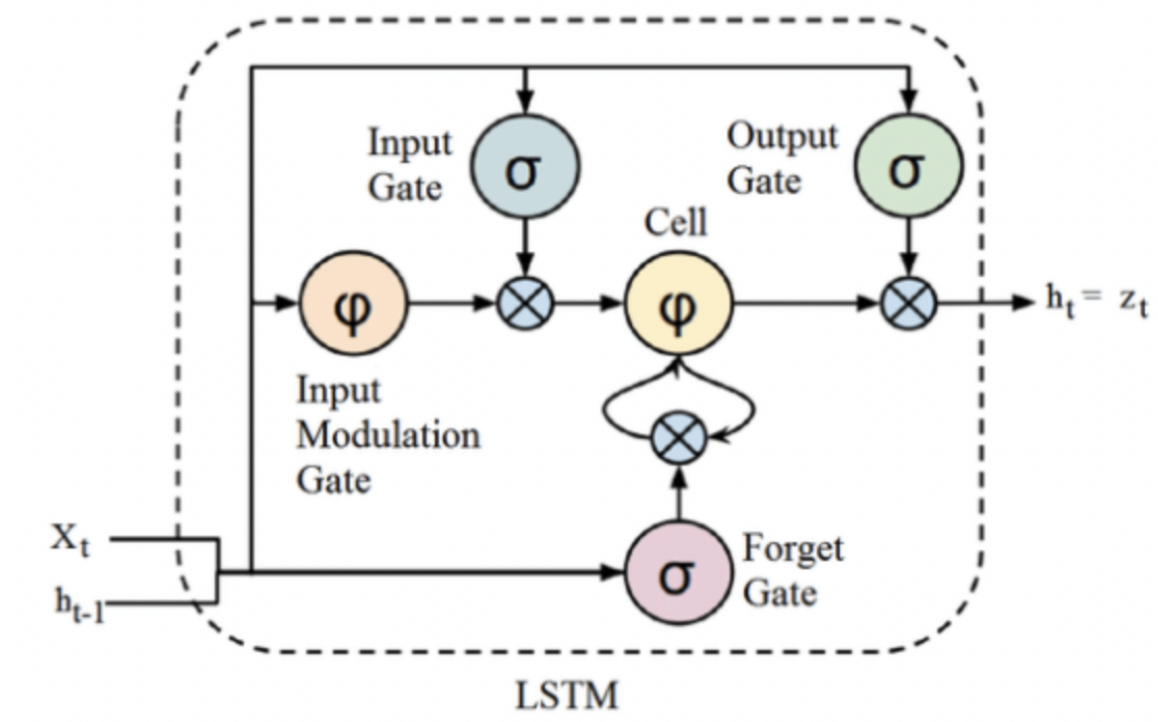

Figura 14 - Gates Rede Neural Recorrente [12]

Dentro da estrutura, existe o conceito de "gates”, que são responsáveis por remover ou adicionar informações ao estado da célula atual, que são as informações guardadas de módulos anteriores. A célula é composta por três gates, são eles: **Input Gate**, **Output Gate** e **Forget Gate**.

Começando pelo **Forget Gate**, temos que ele é o responsável pelo "esquecimento" ou não reutilização dos dados que vem de outra célula. Ele possui duas entradas, sendo a primeira a **xt** (entrada no momento específico) e a segunda a **ht-1** (saída de célula anterior). Essas são alimentadas ao gate e multiplicadas por matrizes de peso, seguidas pela adição do bias. O resultante é passado por uma função de ativação

que fornece uma saída binária. Se para um determinado estado de célula a saída for 0, a informação é esquecida e para a saída 1, a informação é retida para uso futuro. Passando agora para o **Input Gate**, temos que é o responsável por adicionar informações pertinentes às células. Primeiramente, a informação é regulada usando a função sigmóide que filtra os valores a serem lembrados de forma similar ao forget gate usando as entradas **ht-1** e **xt**. Então, um vetor é criado usando a função **tanh** que dá saída de -1 a +1, que contém todos os valores possíveis de **ht-1** e **xt**. Para finalizar, temos os **Output Gates**. Esses são responsáveis por extrair informações úteis do estado da célula atual para serem apresentadas. Para tal ação, um vetor é gerado aplicando a função **tanh** na célula. Então, a informação é regulada usando a função sigmóide que filtra os valores a serem lembrados usando as entradas **ht-1** e **xt**. Os valores do vetor e os valores regulados são multiplicados para serem enviados como uma saída e entrada para a próxima célula.

Abaixo, temos uma sequência de imagens mostrando o processo ocorrendo dentro de uma célula de uma LSTM.

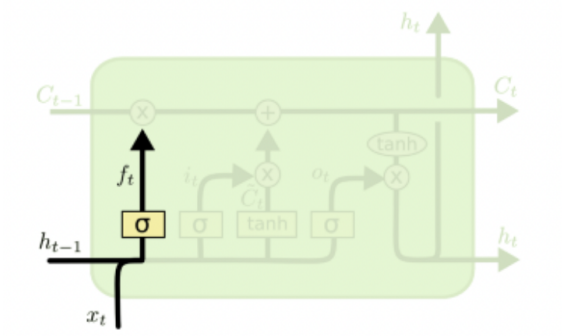

Figura 15 - Forget Gate [11]

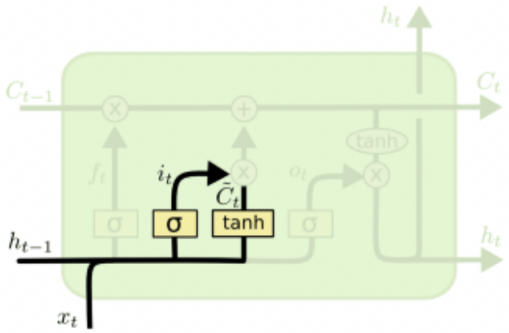

Figura 16 - Input Gate [11]

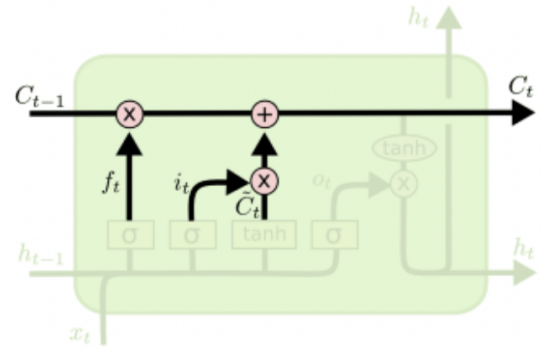

Figura 17 - Atualização de Estado [11]

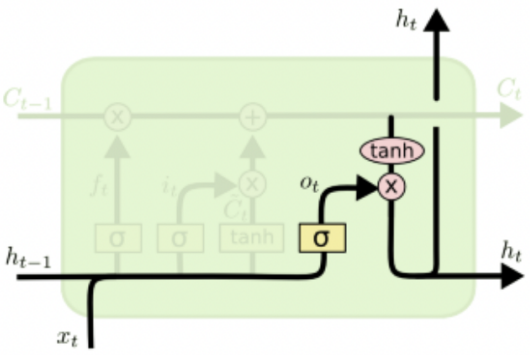

Figura 18 - Output [11]

7 – Vapnik–Chervonenkis(VC) dimensions

Dentro do universo de aprendizado de máquina temos conceitos extremamente importantes quando o assunto é qualidade do nosso modelo. Quando falamos em qualidade, estamos nos referindo a acurácia do modelo criado. Para chegar nessa qualidade precisamos nos atentar a qualidade dos dados que estamos utilizando, bem como a quantidade do mesmo. Quanto mais exemplos tivermos para passar para o nosso modelo, mais inteligente na hora de classificar ele fica. Porém, precisamos, de alguma forma, conseguir mensurar a capacidade do nosso modelo. Para tal, Vapnik-Chervonenkis apresenta sua teoria denominada "Vapnik-Chervonenkis Dimension".

As dimensões VC são usadas para quantificar o quão poderoso é um modelo. Falando de forma prática e intuitiva, a dimensão VC (VC-dim) de um conjunto de funções F, que são nossos classificadores, (VC-dim(F)) é a cardinalidade do maior conjunto de dados que pode ser "cortado" por F. Na imagem abaixo temos as possibilidades de rotulação de três amostras no R^2 e a classificação realizada por uma função linear.

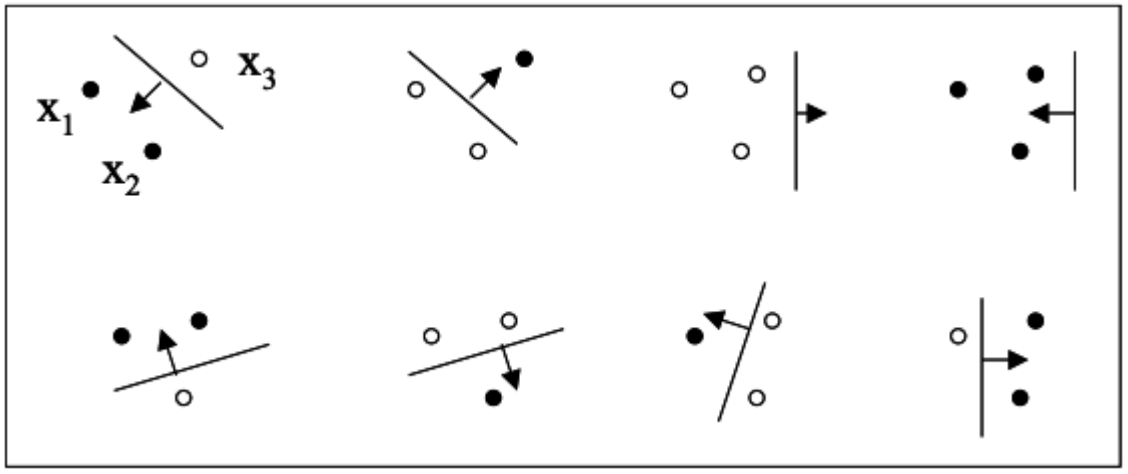

Figura 19 - VC-dimension [4]

Agora, falando em termos matemáticos, a dimensão VC de um espaço de hipóteses H pode ser interpretada como o número efetivo de parâmetros de modelos pertencentes a este espaço de hipóteses H, sendo denotada por h = dVC(H). Quando h = dVC(H) é finita, então modelos pertencentes a H podem generalizar, no sentido de que, tomando-se um conjunto de dados N, com N suficientemente elevado, o risco empírico tende ao risco esperado tanto quanto se queira. Formalmente, h = dVC(H) está

associada ao maior número de pontos que um modelo pertencente a H pode classificar corretamente, para toda proposta de rotulação binária possível desses pontos. Diz-se então que H particiona esses pontos.

Abaixo temos uma tabela com os nossos classificadores e suas respectivas VC-dim.

	Classificador	Dimensão VC
1	SVM	VC = *min*(*dim*(E), $\frac{D^2}{M^2}$)
2	Redes Neurais	VC = *dim*(E) × $\|N\|$
3	Redes Bayesianas	VC ≈ $\#params(T)$
4	Árvores de decisão T	VC ≈ $\#nos(T)$

8 – Resultados

Nesta seção trataremos de falar sobre os resultados obtidos pelos nossos modelos de classificação e regressão. Comentaremos também, em cada um deles, o motivo dos resultados obtidos e o que poderiam melhorar.

8.1 – Modelos de classificação

Para os resultados que obtivemos utilizamos um arquivo de dados exportado pelo Surfguru. Nesse arquivo continham dados desde 01 de Agosto de 2008 até o dia 01 de Agosto de 2021. E, para fazer nossos testes dos modelos e das suas respectivas acurácias, utilizamos o nosso crawley para extrair os dados da mesma fonte, porém, no período de 01 de Agosto de 2021 até 01 de Outubro de 2021.

	Random Forest	Naive Bayes	Logistic Regression	Kernel SVM
Período de criação dos modelos	86.77%	77.32%	66.42%	90.86%

Dos quatro modelos utilizados, tivemos o Kernel SVM como mais preciso, seguido do Random Forest, Naive Bayes e Logistic Regression. Apesar do Kernel SVM e do Random Forest serem os dois mais precisos respectivamente, achamos melhor desconsiderar tamanha precisão uma vez que a probabilidade de ter ocorrido overfitting é muito grande, já que, como vimos anteriormente, suas VC dimension são altas o que indica que eles lidam muito bem com uma grande quantidade de dados, o que foge da nossa realidade visto que temos, dentro do período de criação dos modelos, somente 4990 linhas de informação que correspondem aos dados dos dias desde o início desse período até o fim do mesmo.

Olhando agora para o nosso terceiro colocado em acurácia, temos Naive Bayes que, por sua vez, dentre os modelos que utilizamos, é o que tem a menor VC-dim, logo é o que melhor lidará com poucas quantias de dados, se encaixando perfeitamente para a nossa realidade.

Apesar de não termos considerado o modelo do Random Forest como a nossa diretriz, conseguimos extrair dele algumas informações interessantes relacionadas com os nossos dados. A primeira delas foi a questão da importância das features e quais foram as mais relevantes para o modelo tomar suas decisões. Conversando com a equipe da área de oceanografia, tivemos os inputs de que, com certeza, a direção do vento e da ondulação deveriam estar entre as features mais importantes dos nossos modelos. Podemos ver na figura abaixo que o nosso modelo também notou a importância desses dados e acabou utilizando-os para suas classificações.

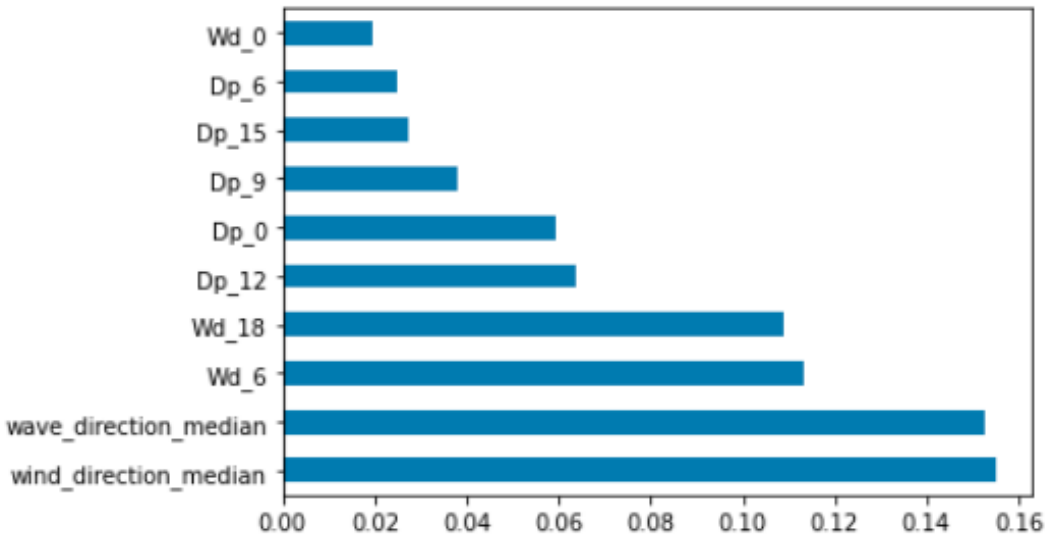

Figura 20 - Features mais relevantes para o Random Forest Classifier

Outra informação interessante que conseguimos através do processamento dos dados e do plot dos mesmos foi a matriz de correlação das features. A matriz de correlação é uma tabela que mostra o quanto uma feature está diretamente interligada com outra e é usada para ter um entendimento prévio dos dados antes de partir para análises mais avançadas. Como podemos ver abaixo, notamos que a direção do vento está muito ligada à qualidade das condições do dia. Apesar de mostrar uma correlação entre features que fazem sentido, às vezes pode ocorrer do nosso modelo fazer correlações que no âmbito real, não fazem sentido estarem correlacionadas, como a questão do período da onda com a direção dela, assim como também não faz sentido ele não fazer um bom relacionamento entre vento e tamanho de onda.

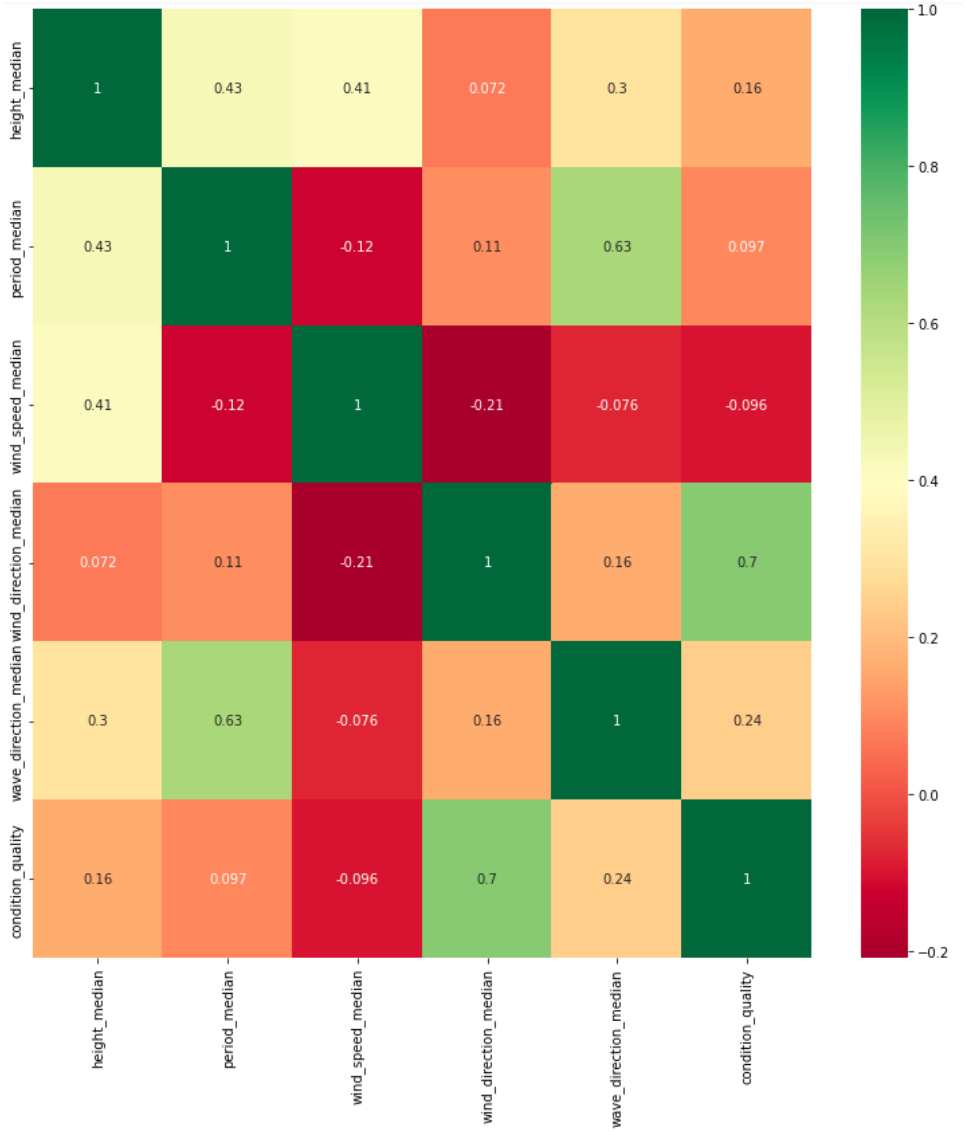

Figura 21 - Matriz de correlação de features

8.2 – Redes Neurais

Como falamos em tópicos mais acima, acabamos por utilizar uma rede neural recorrente LSTM por ser a que mais se encaixava dentro do que estávamos buscando executar, já que, como falamos na parte de LSTM no ponto 6.3.1, ela é bem adequada para classificar, processar e prever séries temporais com intervalos de tempo de duração desconhecida, o que significa que ela "lembra" dos valores desses intervalos arbitrários e utiliza isso para efetuar as previsões futuras. Algumas das aplicações mais comuns para as LSTMs são geração de texto, reconhecimento de escrita e predição de séries temporais, por exemplo, predição de valores de ações no mercado financeiro, sendo essa última a aplicação mais parecida com o que queremos fazer, pois, assim como ela prevê um determinado valor para uma ação baseado nos N indicadores de baixa e alta das ações, estamos querendo, baseado em N indicadores climatológicos e oceanográficos, prever uma condição do mar para a prática do surf.

Utilizamos como método otimizador do aprendizado o Adam, e como função de perda a MSE (Mean Squared Error). Fizemos uma rodada de 100 épocas de treinamento para a rede e obtivemos a precisão de 16.66%. Para ilustrar o que obtivemos como resultado, plotamos a figura abaixo que mostra o mesmo período de teste citado no tópico de classificadores.

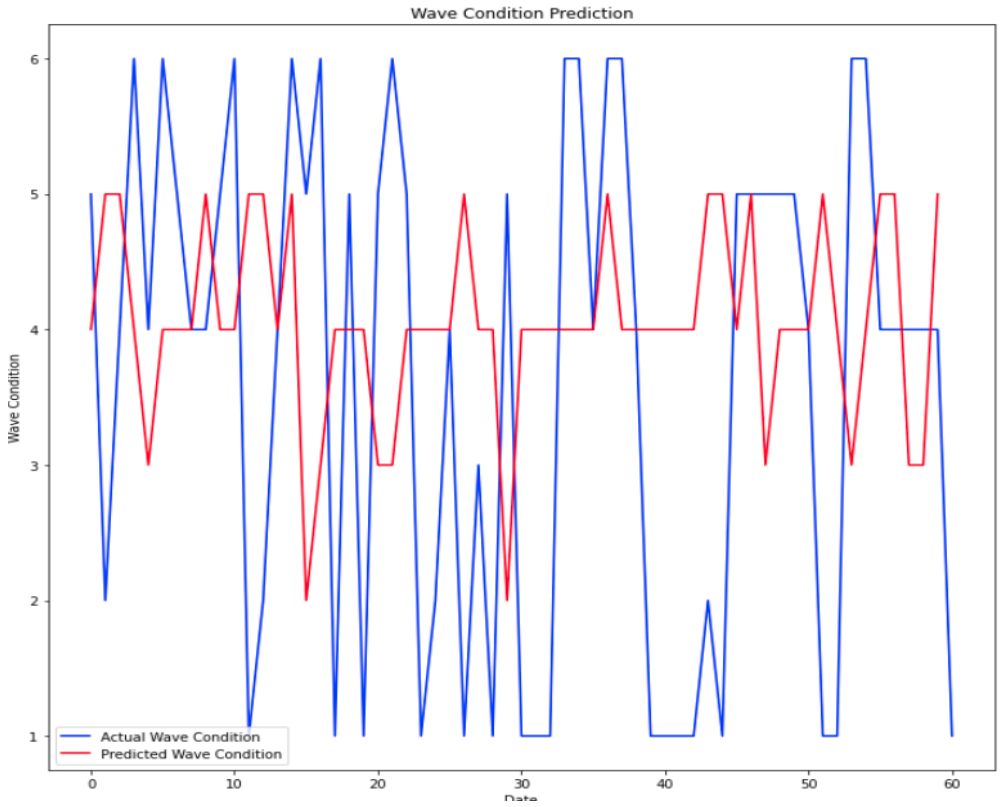

Figura 21 - Gráfico do output da predição da LSTM

Concluímos que, apesar da precisão (que ainda será colocada), o modelo poderia ter uma melhor performance caso tivéssemos mais variações de qualidade de condições estipuladas por nós. Atualmente contamos com somente seis variações de qualidades, o que é muito pouco comparado com o poder que a nossa LSTM pode oferecer a prever.

9 – Considerações Finais

Apesar de termos esperado conseguir fazer o que planejamos no início, não conseguimos executar 100% do cronograma. O tempo que levamos até conseguir dados consistentes fez com que todo o resto planejado atrasasse, uma vez que para cada nova fonte de dados que achávamos tínhamos que fazer todo o estudo e limpeza sobre eles para verificar se seriam válidos para o projeto, impedindo de nos aprofundarmos melhor na criação de números de condições para um pico, que faria com que tivéssemos uma precisão maior no modelo da nossa rede neural LSTM feita no final.

O que mais agregou ao longo da execução desse trabalho foi, além de um maior conhecimento nas técnicas e ferramentas de análise de dados e data science, uma forma de compreender melhor o oceano e a formação das ondas. Muitas coisas que achamos ser de uma forma, se mostraram completamente diferentes, principalmente as relações de alguns fatores climáticos com a formação das ondas, como a correlação entre o período da ondulação com a direção da mesma influenciando mais o modelo do que a velocidade e direção do vento com o tamanho da onda, tendo uma correlação tão alta que quase chegou a ser melhor que a relação da direção do vento com a qualidade da condição.

Provavelmente eu teria focado de forma diferente no desenvolvimento do projeto caso tivesse que fazer ele do zero com todo o conhecimento obtido no desenrolar do mesmo. Com certeza sobraria mais tempo para se preocupar em aperfeiçoar os modelos e tentar criar mais casos de qualidade de condições para um pico de onda, que no caso do projeto foi o de Saquarema, uma vez que já teríamos a fonte de dados do Surfguru e não teríamos que ficar fazendo inúmeras buscas por outras fontes com grandes quantidades de dados e dados consistentes.

Creio que a continuação desse projeto é muito pertinente. A evolução do mesmo teria grande impacto para o cenário da oceanografia pois retiraria o bloqueio que temos, atualmente, de existir somente uma janela de 15 dias para previsões de onda, não tendo

a necessidade de apressar o desenvolvimento de novos modelos matemáticos específicos que ultrapassem essa janela de dias e do esporte mundial pois seria capaz de ser utilizável por grandes organizações tornando os eventos muito mais econômicos e precisos.

10 - Referências bibliográficas

1. **Dados do BNDO**. Diponível em: https://www.marinha.mil.br/chm/dados-do-bndo/acesso-dados-e-produtos
2. Eremenko, K., de Ponteves, H, SuperDataScience Support, Ligency Team, **Machine Learning A-Z: Hands-On Python & R In Data Science**. Disponível em: www.udemy.com/course/machinelearning
3. **Vapnik-Chervonenkis dimension.** Disponível em: Vapnik–Chervonenkis dimension - Wikipedia
4. Von Zuben, Fernando; Boccato, Levy. **Máquinas de Vetores Suporte**. Disponível em: IA353 - Tópico 7 (unicamp.br)
5. Scikit Learn. **Feature importances with a forest of trees**. Disponível em: https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html
6. Navlani, Avnash. **Support Vector Machines with Scikit-learn.** Disponível em: Sklearn SVM (Support Vector Machines) with Python - DataCamp
7. **Surfguru**. https://surfguru.com.br
8. Previsão de Saquarema. **Surfguru previsões**. Disponível em: https://surfguru.com.br/previsao/csv/brasil/rio-de-janeiro/saquarema
9. Floyd, Sally; Warmuth, Manfred. **Sample Compression, Learnability, and the Vapnik-Chervonenkis Dimension**. DIsponível em: Sample compression, learnability, and the Vapnik-Chervonenkis dimension (springer.com)
10. **SiMCosta**. SiMCosta (furg.br)
11. R. F. Junior, Jose. **Redes Neurais Recorrentes — LSTM**. Disponível em: Redes Neurais Recorrentes — LSTM. Os humanos não começam a pensar a… | by Jose R F Junior | Medium

12. Deep Learning Book. **Capítulo 51 - Arquitetura de Redes Neurais Long Short Term Memory**. Disponível em: Capítulo 51 - Arquitetura de Redes Neurais Long Short Term Memory (LSTM) - Deep Learning Book

13. Scikit Learn. **Random Forest Classifier**. Disponível em: sklearn.ensemble.RandomForestClassifier — scikit-learn 1.0.1 documentation

14. **Corporate Finance Institute**. Random Forest - Overview, Modeling Predictions, Advantages (corporatefinanceinstitute.com)

15. Kumar, Vivek**. Random Forest**. Disponível em: Random Forest (towardsmachinelearning.org)

16. Argawal, Animesh**. Polynomial Regression**. Disponível em: Polynomial Regression. This is my third blog in the Machine… | by Animesh Agarwal | Towards Data Science

17. MathWorks. **What is Machine Learning**. Disponível em: What Is Machine Learning? | How It Works, Techniques & Applications - MATLAB & Simulink (mathworks.com)

18. Environmental Modeling Center. **Model Description**. Disponível em: WAVEWATCH III Model Description (noaa.gov)