INSTITUTO TECNOLÓGICO DE AERONÁUTICA

Bruno de Souza Neves

DETERMINAÇÃO DE COEFICIENTES AERODINÂMICOS USANDO MACHINE LEARNING EM PYTHON

Trabalho de Graduação
2019

Curso de Engenharia de Computação

CDU 681.3:16

Bruno de Souza Neves

DETERMINAÇÃO DE COEFICIENTES AERODINÂMICOS USANDO MACHINE LEARNING EM PYTHON

Orientador

Prof.Dr. Paulo Tasinaffo (ITA)

Coorientador

Ten Cel Piterson Marques Lisboa (CCA-SJ)

ENGENHARIA DE COMPUTAÇÃO

São José dos Campos
Instituto Tecnológico de Aeronáutica

2019

Dados Internacionais de Catalogação-na-Publicação (CIP)
Divisão de Informação e Documentação

Neves, Bruno de Souza
Determinação de Coeficientes Aerodinâmicos usando Machine Learning em Python / Bruno de Souza Neves.
São José dos Campos, 2019.
119f.

Trabalho de Graduação – Curso de Engenharia de Computação– Instituto Tecnológico de Aeronáutica, 2019. Orientador: Prof.Dr. Paulo Tasinaffo. Coorientador: Ten Cel Piterson Marques Lisboa.

1. Simulador de Vôo. 2. Coeficientes Aerodinâmicos. 3. TensorFlow. 4. Regressão. 5. Redes Neurais. 6. Python. 7. Keras. I. Instituto Tecnológico de Aeronáutica. II. Título.

REFERÊNCIA BIBLIOGRÁFICA

NEVES, Bruno de Souza. **Determinação de Coeficientes Aerodinâmicos usando Machine Learning em Python**. 2019. 119f. Trabalho de Conclusão de Curso (Graduação) – Instituto Tecnológico de Aeronáutica, São José dos Campos.

CESSÃO DE DIREITOS

Bruno de Souza Neves
Rua H8A, 132
12.228-460 – São José dos Campos–SP

DETERMINAÇÃO DE COEFICIENTES AERODINÂMICOS USANDO MACHINE LEARNING EM PYTHON

Essa publicação foi aceita como Relatório Final de Trabalho de Graduação

Bruno de Souza Neves

Autor

Prof. Dr. Paulo Marcelo Tasinaffo

Orientador

Prof. Inaldo Capistrano Costa

Coordenador do Curso de Engenharia de Computação

São José dos Campos, 29 de novembro de 2019

Dedico este trabalho a todos aqueles que se dispõem a contribuir com o desenvolvimento desse gigante que é o Brasil.

Agradecimentos

Primeiramente , gostaria de agradecer a Deus pela vida que tive até então , que mesmo com todas as adversidades, dificuldades e incertezas, pude ser guiado para o caminho certo, mesmo errando incontáveis vezes.

A minha Mãe, por nunca duvidar de mim e sempre ter me dado o apoio em qualquer situação que seja.

Ao meu segundo pai , Rogério, por ter me dado conselhos , suporte e conselhos valiosos.

A minha irmã, que mesmo com apenas 7 anos, me incentiva a ser melhor a cada dia.

Ao meu pai , pelo amor e confiança.

Ao Ten.Cel. Piterson, por ser uma pessoa compreensível e que forneceu grande ajuda quanto à transmissão de conhecimentos sobre Aerodinâmica.

Ao Prof. Dr. Paulo Marcelo Tasinaffo, pelas aulas ministradas no ITA

Aos amigos que fiz no ITA ,por tantas memórias que ecoarão na minha cabeça até meus últimos dias de vida.

"Do what nobody does to achieve what nobody has'"'
— Chris Heria

Resumo

O trabalho propõe-se a desenvolver um modelo em linguagem Python que faça predições confiáveis e ágeis dos coeficientes aerodinâmicos da aeronave EMB-312 TUCANO. Tais coeficientes são necessários para reprodução de modelos aerodinâmicos a serem usados em simuladores de vôo.

As abordagens principais para a resolução do problema foram obtidas utilizando Redes Neurais(TensorFlow 2.0) e Random Forests , sendo implementadas em linguagem Python®. Os dados de treinamento para as redes são derivados de ensaios em túnel de vento e os coeficientes são modelados como funções adimensionais de características do escoamento, pressão dinâmica, e das deflexões superfícies de controle da aeronave.

Esse estudo tem o intuito de diminuir a quantidade de ponto tomados em uma campanha de ensaio em túnel, diminuindo os custos operacionais para desenvolvimento de um modelo aerodinâmico preciso. É feita então um benchmarking com osmétodos de aprendizado disponívies em Python para melhor modelagem aerodinâmica a ser usada em simuladores de vôo.

Abstract

This work proposes to develop a model in Python that makes reliable and fast predictions of the aerodynamic coefficients of the EMB-312 TUCANO. Such coefficients are necessary for modeling aerodynamic models that are going to be used in fligh simulators.

The main approaches to the problem were obtained using Neural Networks (TensorFlow 2.0) in Python® and Random Forests Predictors. The training data for the networks were derived from wind tunnel tests and the coefficients were modeled as functions of the characteristics of the flow and the control surfaces of the aircraft.

This study aims to reduce the number of points taken in a wind-tunnel campaign, reducing operating costs for the development of a precise aerodynamic model. A Benchmarking of Machine Learning methods avaiable in Python is made so that the best model is used in fligh simulators.

Lista de Figuras

Lista de Símbolos

$\bar{c}$	Corda Média Aerodinâmica
F	Força
L	Força Aerodinâmica de Sustentação
D	Força Aerodinâmica de Arrasto
Y	Força Aerodinâmica Lateral
l	Momento Aerodinâmico de Rolamento
m	Momento Aerodinâmico de Arfagem
n	Momento Aerodinâmico de Guinada
Re	Número de Reynolds
M_∞	Número de Mach do escoamento
c	Comprimento de Referência
q	Pressão Dinâmica
S	Área de Referência
b	Envergadura
C_D	Coeficiente de Arrasto
C_L	Coeficiente de Sustentação
C_Y	Coeficiente de Força Lateral
C_l	Coeficiente de Momento de Rolamento ou Giro
C_m	Coeficiente de Momento de Arfagem ou de Inclinação ou Picador
C_n	Coeficiente de Momento de Guinada
α	Ângulo de Ataque em graus
β	Ângulo de Derrapagem em graus
δ_f	Deflexão do Flape em graus
δ_e	Deflexão do Profundor em graus
δ_a	Deflexão do Aileron em graus
δ_{ae}	Deflexão do Aileron Esquerdo em graus
δ_{ad}	Deflexão do Aileron Direito em graus
δ_r	Deflexão do Leme em graus
$\Re$	Conjunto dos Reais
$x_j^{(i)}$	Valor da variável j no i-ésimo exemplo de treino

$x^{(i)}$	Vetor coluna de de todas as variáveis do i-ésimo exemplo de treino
x^T	Vetor transposto de x
$a_i^{(j)}$	Ativação da unidade i na camada j
$z_i^{(j)}$	Hipótese intermediária de uma função g da variável i na camada j
$\Theta^{(j)}$	Matriz de pesos controlando o mapeamento da camada j para $j+1$

1 Introdução

Machine Learning é a ciência (e arte) de programas de computadores aprenderem com dados.

Em 1959, Arthur Samuel definiu aprendizado de máquina como o "campo de estudo que dá aos computadores a capacidade de aprender sem serem explicitamente programados". Uma versão mais orientada a engenharia seria : "Um programa de computador é dito aprender de uma experiência E com relação a alguma tarefa T e alguma métrica de desempenho P se, se tal performance T , medida por P, melhora com a experiência E", definição de Mitchell.

Machine Learning é interessante no caso de modelos aerodinâmicos, dado que a maior parte dos problemas encontrados na área de dinâmica de simulação de voo estão relacionadas ao comportamento não-linear dos coeficientes aerodinâmicos e a quantidade de dados necessários para estudo e modelagem de um comportamento aeronáutico confiável.

1.1 Motivação

Modelos Aerodinâmicos confiáveis são essenciais para que simuladores de voo reproduzam com fidelidade atividades operacionais reais. Para que tais modelos sejam confiáveis, os parâmetros que os definem precisam ser também confiáveis. A aquisição dos dados são fornecidas por ensaios de túnel de vento, a questão é que tal aquisição é custosa demais para uma tarefa que pode ser resolvida de outra maneira, que é a motivação deste trabalho.

1.2 Objetivo

Por meio de dados experimentais obtidos em túnel de vento com modelo de escala reduzida, determinar os coeficientes aerodinâmicos da aeronave EMB-312 TUCANO em diversas configurações para futura implementação do modelo aerodinâmico no simulador de voo do Centro de Computação da Aeronáutica de São José dos Campos.

2 Aquisição dos dados

Nesta seção do relatório é mostrado como foram adquiridos os dados usados para posterior treinamento . Os ensaios são feitos de formar a simular diferente estados aos quais a aeronave poderá se encontrar. Tais estados são caracterizados pelas seguintes entradas : flapes, ailerons, leme e profundor, e fixando se esses parâmetros , é feita uma varredura em outras duas entradas , α e β . A partir disso, foram calculados os coeficientes aerodinâmicos.

2.1 Dados do modelo

2.1.1 O modelo e Medidas de Referência

No relatório do ensaio (SAMPAIO, 1985), o modelo, representado na figura 2.1 com a convenção de forças e momentos, tem a fuselagem moldada em fibra de vidro com estrutura interna de aço; a asa é em aço e recoberta com resina; o trem de pouso é em aço e alumínio; as empenagens têm estrutura de alumínio revestida com resina e as superfícies de controle (flapes, ailerons, leme e profundor) são de alumínio.

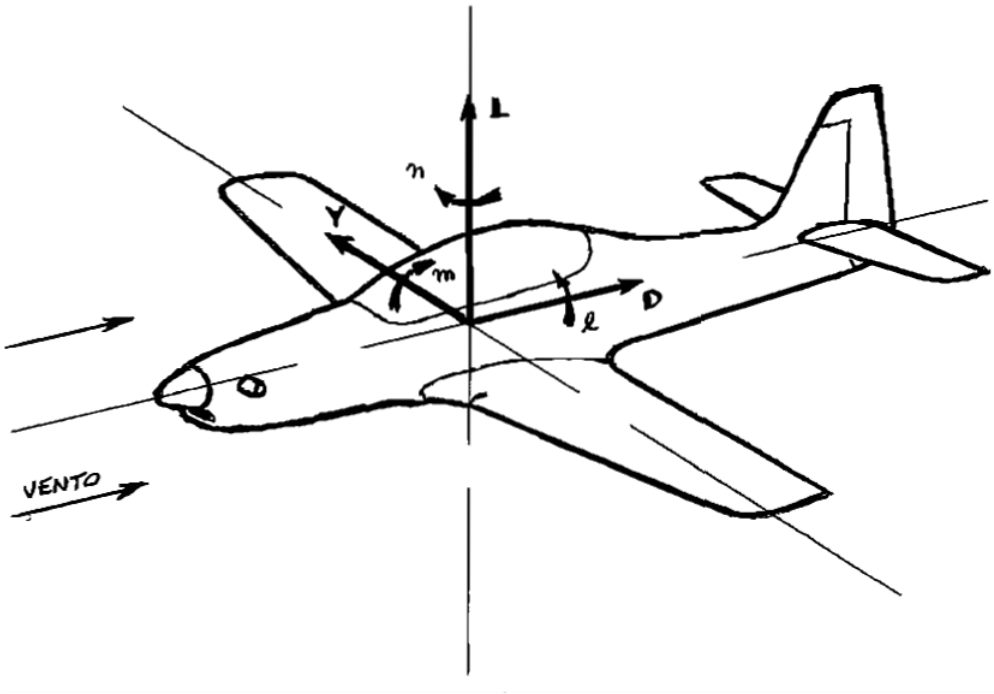


FIGURA 2.1 – Modelo da aeronave EMB-312 (SAMPAIO, 1985).

2.1.2 Sistema de eixos e ângulos

Os coeficientes aerodinâmicos foram aferidos em três diferentes sistemas de eixos, caracterizados pela figura 2.2:

- sistema do vento
- sistema de estabilidade
- sistema de corpo

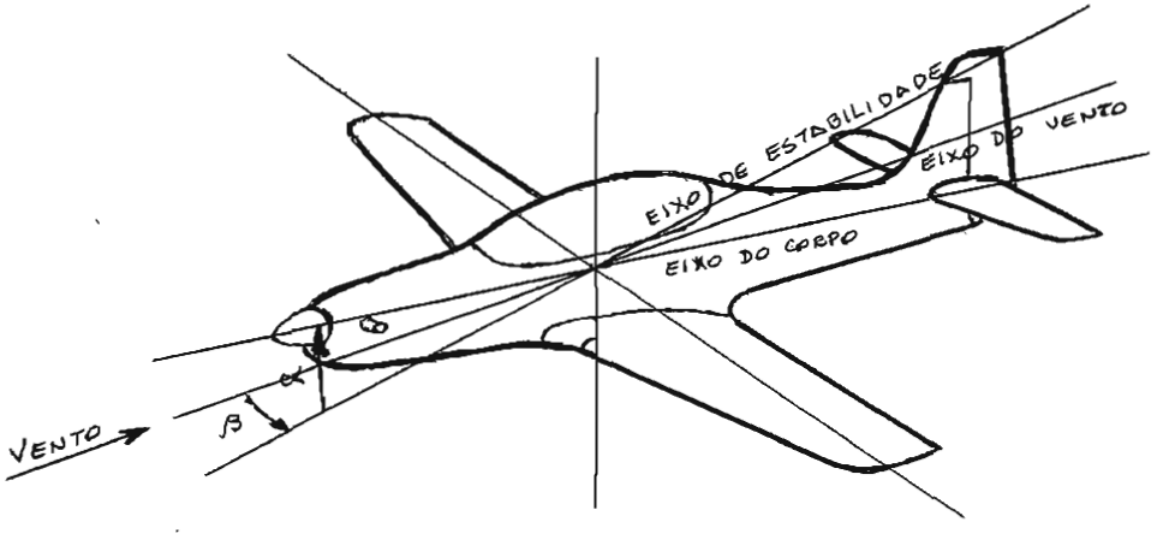


FIGURA 2.2 – Modelo da aeronave EMB-312 e a convenção de eixos adotada, retirada de (SAMPAIO, 1985).

Destaca-se na figura 2.4, a convenção adotada de ângulos para o ensaio.

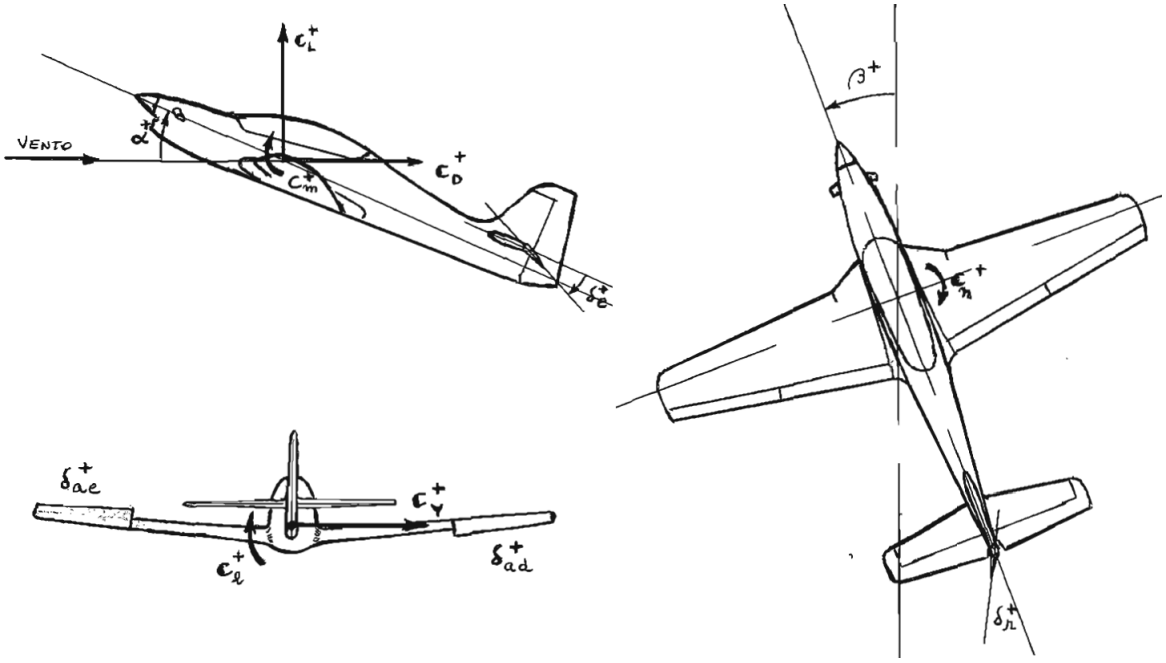


FIGURA 2.3 – Modelo da aeronave EMB-312 e a convenção de ângulos adotada, retirada de (SAMPAIO, 1985).

2.2 Metodologia do Ensaio

Colocada a aeronave na posição de ensaio, a carga atuante no ponto de referência da balança (ponto ”zero”) é medida por seis sensores de carga (”load cells”), três de força e três de momentos, que emitem sinais elétricos proporcionais às cargas atuantes. Os sinais dos sensores, amplificados e filtrados, são levados para um sistema automático de leitura de dados.

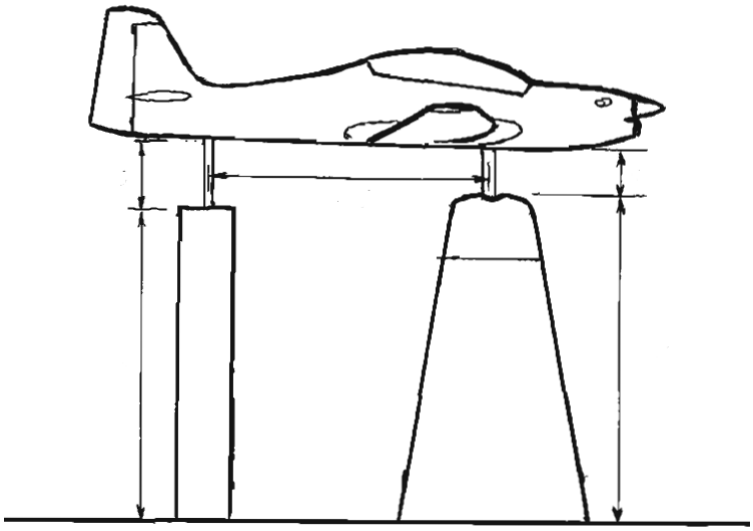


FIGURA 2.4 – Modelo da aeronave EMB-312 na posição de ensaio, retirada de (SAMPAIO, 1985).

A saída do sensor era lida 30 vezes por ensaio. O resultado final apresentado é a média das e leituras. Os valores médios de cada sensor eram, então, reduzidos nos valores dos coeficientes de força e momentos.

2.3 Resultados do Ensaio

2.3.1 Tabela dos Coeficientes Aerodinâmicos

As entradas geradas pelo relatório de ensaios (Figura 2.5) são :

- Ângulo do flape
- Ângulo dos ailerons
- Posição do trem de pouso
- Ângulo do profundor
- Ângulo de ataque corrigido (α)
- Ângulo de guinada corrigido (β)

Com as seguintes saídas

- Coeficiente de Arrasto (C_D)
- Coeficiente de Sustentação (C_L)
- Coeficiente de Momento de Arfagem (C_m)
- Coeficiente de Força Lateral (C_Y)
- Coeficiente de Momento de Rolamento (C_l)
- Coeficiente de Momento de Guinada (C_n)

Sendo os valores da primeira linha referentes ao eixo do vento, na segunda ao eixo de estabilidade e, por fim, na terceira linha ao eixo do avião.

59101	0.2	0.196	0.369	-0.149	0.166	0.007	-0.076	281.9
	-30.0	0.004	0.369	-0.151	0.192	-0.006	-0.076	
		0.002	0.369	-0.151	0.192	-0.006	-0.076	
59102	0.3	0.145	0.403	-0.131	0.142	0.005	-0.067	282.6
	-25.0	0.012	0.403	-0.132	0.160	-0.004	-0.067	
		0.010	0.403	-0.132	0.160	-0.004	-0.067	
59103	0.3	0.106	0.405	-0.089	0.111	-0.007	-0.054	281.0
	-20.0	0.024	0.405	-0.069	0.123	-0.011	-0.054	
		0.022	0.405	-0.069	0.123	-0.011	-0.054	
59104	0.2	0.078	0.389	-0.043	0.082	-0.017	-0.040	282.2
	-15.0	0.033	0.389	-0.016	0.089	-0.018	-0.040	
		0.031	0.389	-0.016	0.089	-0.018	-0.040	

FIGURA 2.5 – Exemplo do modelo de tabela de coeficientes aerodinâmicos gerado pelo ensaio, retirada de (SAMPAIO, 1985).

3 Fundamentação Teórica

3.1 Fundamentos de Aerodinâmica

3.1.1 Forças e Momentos Aerodinâmicos

Independente da complexidade da forma de um corpo, a geração de forças e momentos aerodinâmicos nele são resultado de duas fontes principais: a distribuição de pressão e a tensão de cisalhamento sobre a superfície do corpo. A primeira atua normal à superfície, enquanto o segundo atua tangencialmente à superfície - resultado do atrito causado entre o corpo e o fluido.

Pela teoria da mecânica clássica, o efeito da distribuição de pressão e tensão de cisalhamento, pode ser integrado sobre toda a superfície do corpo, obtendo uma força e um momento aerodinâmicos no corpo. Definida a direção do escoamento, podemos decompor a força resultante em: Sustentação e Arrasto. A primeira é perpendicular ao vetor velocidade do escoamento e a segunda paralela.

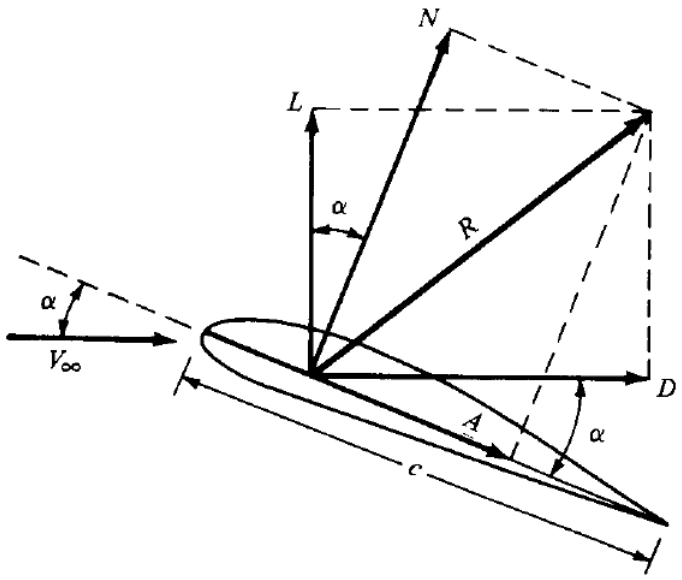


FIGURA 3.1 – Força resultante e suas componentes, retirada de (ANDERSON, 2001).

3.1.2 Coeficientes Aerodinâmicos

Os coeficientes aerodinâmicos são o objeto central do trabalho. Eles serão as saídas dos modelos a serem programados. Eles dependem basicamente das características geométricas do corpo e de características do escoamento, como viscosidade

A seguir são fornecidas as equações que definem as saídas.

- C_L: medida da força aerodinâmica que é perpendicular à direção do vento e oposta à ação da gravidade. É normalmente determinado em ensaios de túnel de vento com boa precisão para a maioria das aeronaves.

$$C_L = \frac{L}{qS} \tag{3.1}$$

- C_D: medida da força aerodinâmica que se opõe ao movimento do corpo. Neste coeficiente, devemos considerar características complexas do escoamento para determinar a representatividade do modelo, por exemplo: efeitos de viscosidade e compressibilidade.

$$C_D = \frac{D}{qS} \tag{3.2}$$

- C_Y: medida da força aerodinâmica que atua perpendicularmente ao eixo do corpo.

$$C_Y = \frac{Y}{2qS} \tag{3.3}$$

- C_m: medida do momento aerodinâmico característico da rotação do nariz do avião "para cima e para baixo" no eixo y, no movimento conhecido como picador (em inglês, *pitch*) e controlado pelo profundor.

$$C_m = \frac{m}{qS\bar{c}} \tag{3.4}$$

- C_l: medida do momento aerodinâmico característico da rotação da ponta de asa no eixo x, conhecido como rolamento (em inglês, *roll*) e controlado pelos ailerons.

$$C_l = \frac{l}{qSb} \tag{3.5}$$

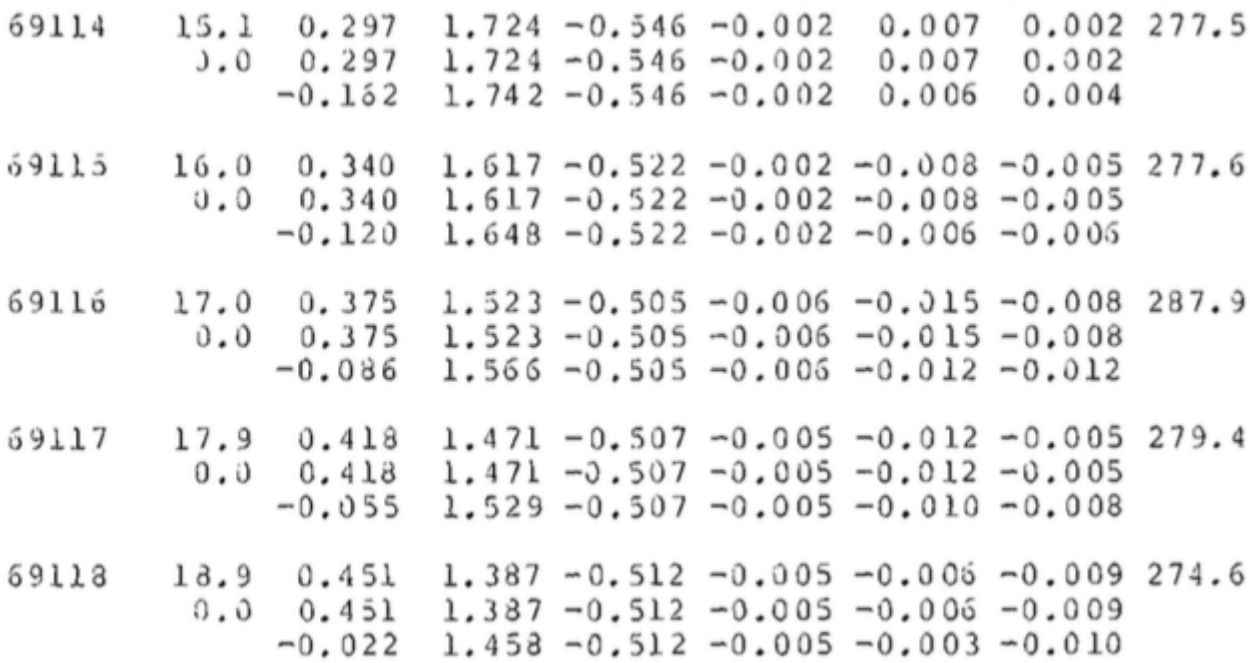
- C_n: medida do momento aerodinâmico característico da rotação do nariz do avião "para os lados" no eixo z, no movimento conhecido como guinada (em inglês, *yaw*) e controlado pelo leme.

$$C_n = \frac{n}{qSb} \tag{3.6}$$

3.2 Método de Interpolação

Os dados fornecidos nas tabelas de ensaio em voo não cobrem todo o espectro para as possíveis outras entradas para determinação dos coeficientes (por exemplo, coeficiente

de sustentação para um valor de 0,7 graus para o ângulo de ataque). A interpolação é uma técnica clássica amplamente usado em simuladores de voo para se obter um espectro contínuo de entradas.



69114	15.1	0.297	1.724	-0.546	-0.002	0.007	0.002 277.5
	0.0	0.297	1.724	-0.546	-0.002	0.007	0.002
		-0.162	1.742	-0.546	-0.002	0.006	0.004
69115	16.0	0.340	1.617	-0.522	-0.002	-0.008	-0.005 277.6
	0.0	0.340	1.617	-0.522	-0.002	-0.008	-0.005
		-0.120	1.648	-0.522	-0.002	-0.006	-0.006
69116	17.0	0.375	1.523	-0.505	-0.006	-0.015	-0.008 287.9
	0.0	0.375	1.523	-0.505	-0.006	-0.015	-0.008
		-0.086	1.566	-0.505	-0.006	-0.012	-0.012
69117	17.9	0.418	1.471	-0.507	-0.005	-0.012	-0.005 279.4
	0.0	0.418	1.471	-0.507	-0.005	-0.012	-0.005
		-0.055	1.529	-0.507	-0.005	-0.010	-0.008
69118	18.9	0.451	1.387	-0.512	-0.005	-0.006	-0.009 274.6
	0.0	0.451	1.387	-0.512	-0.005	-0.006	-0.009
		-0.022	1.458	-0.512	-0.005	-0.003	-0.010

FIGURA 3.2 – O mapeamento entre ângulo de ataque e coeficientes aerodinâmicos é discreta . A interpolação é um dos métodos para obter um espectro contínuo.

3.3 Análise estatística

Métricas estatísticas são necessárias para correta avaliação dos dados. Algumas métricas são :

3.3.1 Valor Médio

O valor médio de uma grandeza x qualquer, medido N vezes é dado por

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{3.7}$$

3.3.2 Mediana

A mediana é o valor que separa a metade maior da menor de uma amostra. Ela é extremamente útil para vericação de pontos atípicos, também chamados de *outliers*, pois identifica valores típicos da amostra (tendência central), sem ser distorcida por valores

muito altos ou muito baixos, como acontece com a média.

3.3.3 Desvio Padrão

O desvio padrão mede a dispersão de um conjunto de dados. Para um *Dataset* com N entradas , temos :

$$\sigma_x = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2} \tag{3.8}$$

3.3.4 Correlação

A correlação é a relação estatística entre duas variáveis. Neste trabalho, o grau de correlação é medido pelo coeciente de *Pearson* apenas para análise qualitativa, já que a medida não representa, necessariamente, uma relação causal.

$$\rho_{xy} = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sigma_x \sigma_y} \tag{3.9}$$

O valor de ρ varia no intervalo entre -1 e +1 e é classicado de modo que a relação entre as variáveis seja : correlação desprezível se $\rho = 0$ e forte se $\rho = 1$.

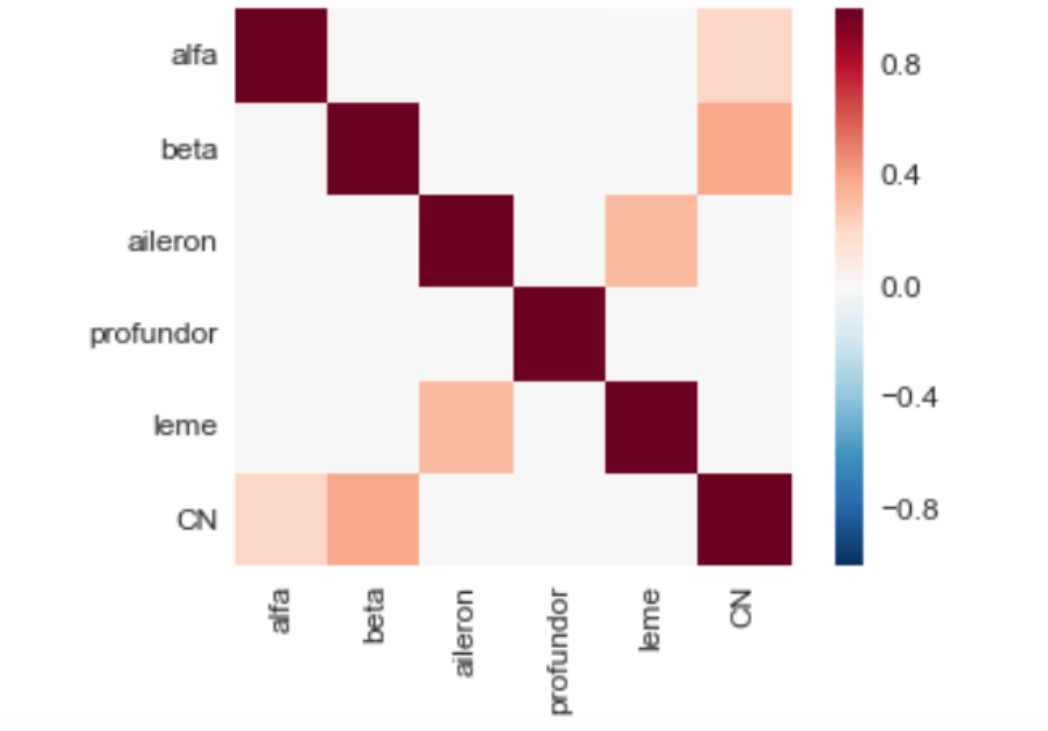


FIGURA 3.3 – Correlação entre entradas e coeficiente de arrasto

3.3.5 Interquartil

Para tratamento de *outliers*, será utilizado o método interquartil .Durante a coleta de dados, erros nos sensores são comuns e precisamos saber como tratá-los, além de erros provenientes da conversão dos dados em *.png* para *.csv*. Assim, é definido um critério para validar os dados em uma amostra de limite superior e inferior (*boxplot*) nos quartis e todos os dados fora desse intervalo são descartados, como a seguir: 3.10 e 3.11, sendo c uma constante em $\Re$.

$$LI_x = \tilde{X_m} - c(\tilde{X_M} - \tilde{X_m}) \tag{3.10}$$

$$LS_x = \tilde{X_M} + c(\tilde{X_M} - \tilde{X_m}) \tag{3.11}$$

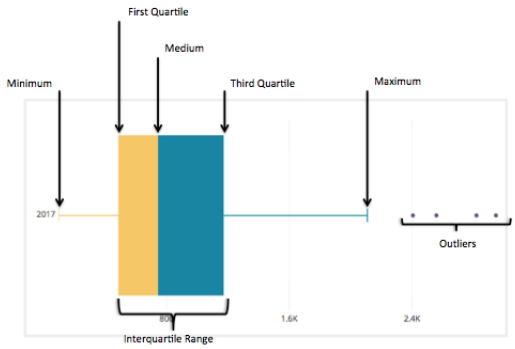


FIGURA 3.4 – Exemplo de *boxplot*.

Para o trabalho $c = 3$ mostrou-se adequado para tratamento dos *outliers* .

3.3.6 Comparação dos Erros de Predição

É necessário uma métrica para comparação dos modelos. No trabalho as seguintes grandezas serão calculadas para comparação dos modelos em *Random Forest* e da Rede Neural:

Mean squared error	$\text{MSE} = \frac{1}{n}\sum_{t=1}^{n} e_t^2$
Root mean squared error	$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$
Mean absolute error	$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n} \lvert e_t \rvert$
Mean absolute percentage error	$\text{MAPE} = \frac{100\%}{n}\sum_{t=1}^{n} \left\lvert \frac{e_t}{y_t} \right\rvert$

3.3.7 Significância Estatística

Um evento observado é considerado estatisticamente significativo quando é altamente improvável que o evento tenha acontecido por acaso. Mais especificamente, um evento observado é estatisticamente significativo quando seu *p value* cai abaixo de um determinado limite, chamado nível de significância. Ultrapassar esse limiar e alcançar significância estatística geralmente marca uma decisão ou conclusão a ser tirada dos resultados de um estudo.

O *p value* é uma probabilidade tão ou mais extrema que um evento observado ocorra. Essa probabilidade também vem com a suposição de que eventos extremos ocorrem com a mesma frequência relativa que ocorrem em circunstâncias normais. Ou seja, um *p value* pode ser considerado uma medida de quão incomum é um evento observado. Quanto menor seu valor, mais incomum é o evento.

3.3.8 Bootstrap Sampling

A ideia básica do *Bootstrap* é realizar inferências acerca de um parâmetro θ (a média de uma população por exemplo). É um método de amostragem com reposição desta maneira diferentes amostragens enfatizam diferentes aspectos de um conjunto de dados. Tal técnica desempenhou um papel importante no trabalho tanto para cálculos de *p value* quanto para o treinamento da *Random Forest*.

3.4 Processamento de Imagem

3.4.1 Kernel

Em processamento de imagens, um Kernel ou matriz de Convolução é uma pequena matriz usada para obter a partir de uma imagem, uma versão com efeitos de *blurring*, *sharpening*, *embossing*, dentre outros da mesma . Em linguagem matemática, sendo f a imagem original (uma matriz de pixels) e ω a matriz de Convolução obtemos a imagem processada F como dado a seguir :

$$F(x,y) = \omega * f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} \omega(s,t) f(x-s, y-t) \tag{3.12}$$

No presente trabalho tal teoria desempenhou um importante papel na obtenção dos dados originais em formato *.png* para formato *.csv* como será abordado mais à frente. A

matriz de Convolução usada no trabalho, por motivos que serão explicitados adiante foi:

$$\omega = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

(3.13)

3.4.2 Open Computer Vision

OpenCV (*Open Source Computer Vision*) é uma biblioteca de programação de código aberto e inicialmente desenvolvida pela Intel com o objetivo de tornar a visão computacional mais acessível a desenvolvedores. Possui diversos métodos de manipulação de imagens que foram explorados para obtenção de imagens com melhor resolução, além de ter uma implementação do método de convolução .

3.4.3 Tesseract

O PyTesseract é uma ferramenta de OCR (*Optical Character Recognition*) em *Python* para reconhecimento de texto em imagens. Tal ferramenta foi usada neste trabalho para conversão de texto em formato *png* para formato *csv* a partir das imagens pré-processadas com OpenCV.

3.4.4 Cinemática de Voo em Simuladores

Equações diferenciais ocorrem frequentemente em engenharia. Se a taxa de variação de uma variável depende de outras variáveis ou da taxa de variação de outras variáveis (suas derivadas), essas equações podem ser expressas como um conjunto de equações diferenciais. Se um sistema pode ser descrito por um conjunto de equações diferenciais, a análise dessas equações poderá prever uma solução exata do comportamento do sistema ou fornecer uma visão do comportamento esperado. Por exemplo, um circuito elétrico que consiste em resistores, capacitores e indutores pode ser expresso como um conjunto de equações diferenciais. Se um sinal é aplicado ao circuito, o ganho, a fase e a estabilidade resultantes do circuito podem ser determinados a partir da inspeção dessas equações. Uma resposta específica a uma onda quadrada ou senoidal pode ser prevista. Portanto, não é de surpreender que equações diferenciais dominem a dinâmica de voo. Obviamente, as entradas do piloto são imprevisíveis - elas dependem das informações observadas pelo piloto

e dos processos mentais e físicos do piloto. No entanto, certas entradas, particularmente uma entrada de impulso podem ser representadas em termos matemáticos e fornecer informações sobre o comportamento ou as qualidades de manuseio de uma aeronave. Se a resposta for muito rápida, a aeronave será ágil e manobrável, mas também pode ser muito exigente voar na presença de distúrbios. Se a resposta for lenta, a aeronave é lenta para responder às demandas do piloto e seria difícil de voar em situações em que mudanças rápidas na trajetória de voo da aeronave são necessárias por exemplo, nos estágios finais de uma aterrissagem. Ao entender o comportamento da aeronave em voo reto e nivelado, em voo de subida e descida e em resposta às entradas do piloto, é possível projetar uma aeronave com características de manuseio conhecidas e desejáveis ou projetar um sistema de controle para garantir qualidades específicas de manuseio. Ao se formular as equações de movimento de uma aeronave, a dinâmica longitudinal e lateral de uma aeronave pode ser simulada modelando as equações diferenciais subjacentes. Essas equações, que dependem dos coeficientes aerodinâmicos, são dadas a seguir :

$$\dot{u} = f_u(F_u, m, \theta, \phi, \psi) \tag{3.14}$$

$$\dot{v} = f_v(F_v, m, \theta, \phi, \psi) \tag{3.15}$$

$$\dot{w} = f_w(F_w, m, \theta, \phi, \psi) \tag{3.16}$$

$$u = \int \dot{u} dt \tag{3.17}$$

$$v = \int \dot{v} dt \tag{3.18}$$

$$w = \int \dot{w} dt \tag{3.19}$$

$$\dot{p} = f_p(M_p, m, \theta, \phi, \psi) \tag{3.20}$$

$$\dot{q} = f_q(M_q, m, \theta, \phi, \psi) \tag{3.21}$$

$$\dot{r} = f_r(M_r, m, \theta, \phi, \psi) \tag{3.22}$$

$$p = \int \dot{p} dt \tag{3.23}$$

$$q = \int \dot{q} dt \tag{3.24}$$

$$r = \int \dot{r} dt \tag{3.25}$$

As acelerações lineares e angulares são calculadas nos eixos da aeronave, com a origem no centro de gravidade. As direções ortogonais das componentes u, v e w são para a frente (para o nariz), estibordo (ao longo da asa direita) e para baixo (perpendicular às asas), respectivamente. Da mesma forma, as componentes p, q e r se referem ao movimento angular sobre esses eixos. As funções f são usadas para calcular as acelerações lineares e angulares das forças F e momentos M e dependem das dimensões da aeronave, massa m, inércias J, entradas de controle, densidade do ar, velocidade do ar, número do Mach, ângulos de incidência e assim por diante. A massa e as inércias podem ser calculadas a partir do peso da aeronave, combustível, passageiros e dimensões da aeronave. Os três ângulos de orientação em relação ao eixo da aeronave são: inclinação, rotação e guinada (θ, ϕ e ψ).

No presente trabalho , as integrações de tais equações a partir de determinadas condições iniciais segundo um método Runge Kutta de ordem 4 foram necessárias para comparação do tempo de execução dos modelos a serem trabalhados. Como será abordado no Capítulo 7 , utilizou-se uma ferramenta *open source* (PyFME) que inclui as equações e métodos de integração necessários para a simulação de dinâmica de voo.

3.5 O que é Aprendizado de Máquina?

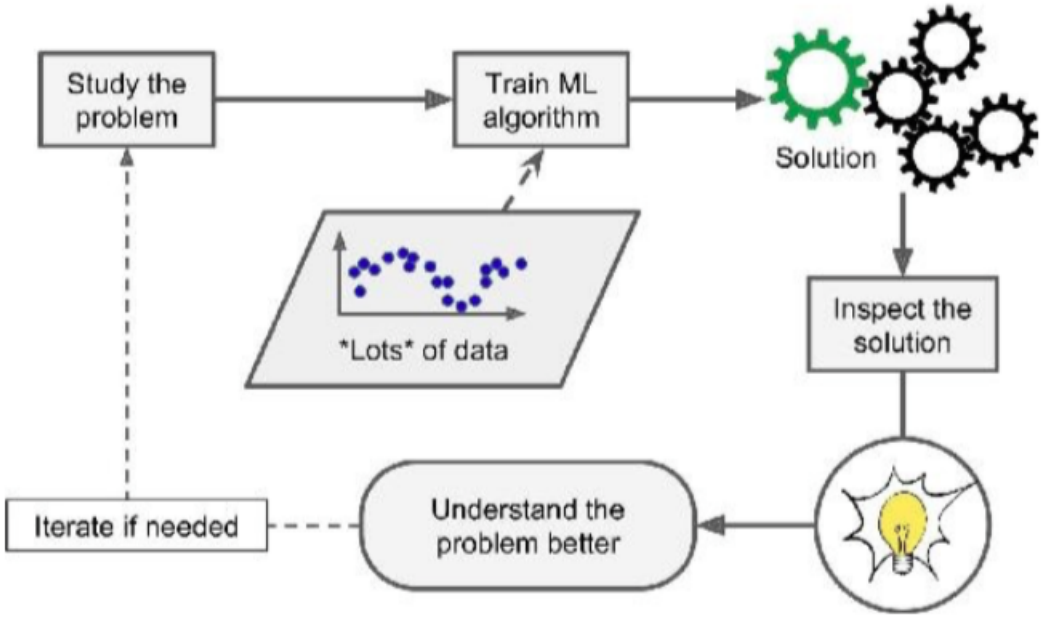


FIGURA 3.5 – Estrutura de um Projeto de Machine Learning (Aurélien - 2018)

3.5.1 Aprendizado Supervisionado

Dado um conjunto de dados de treinamento e um conjunto de saída , o algoritmo aprende com as entradas as saídas desejadas (*labels*). Diferentemente da metodologia tradicional de computação em que a partir de *inputs* e de um programa o objetivo é obter saídas corretas para o fim a que o programa se destina. Em aprendizagem de máquina, o objetivo é , dados o *inputs* e *outputs* de um determinado contexto, o objetivo é a obtenção de um programa que mapeie corretamente ou com menor erro possível , os *inputs* com seus *outputs*. No trabalho tal mapeamento é a regressão,ou seja, queremos mapear os *inputs* das variáveis da aeronave com seus respectivos coeficientes aerodinâmicos,com menor erro possível, já que tratam-se de domínios contínuos .

3.5.2 Ferramentas

TensorFlow é uma biblioteca desenvolvida pelo *Google* para aplicações científicas com foco na estruturação de Redes Neurais por meio de Tensores e operações sobre eles. Após a estruturação do Grafo e inicialização dos conjuntos de treino e validação, executa-se o treinamento por meio de um Sessão,ou seja,execução e estruturação são etapas distintas no *TensorFlow*. Essa biblioteca torna fácil ao programador a customização de hiper parâmetros que minimizem a função custo. No presente trabalho a versão da ferramenta foi a 2.0

,lançada no ano de 2019,com mudanças significativas que auxiliaram no desenvolvimento do trabalho, com melhor sintaxe para treinamento de Rede Neurais.

3.6 Função de Custo

A função de custo é necessária para penalizar erros que o algoritmos cometa durante treinamento. Dependendo de cada método, a função de custo muda para que a penalidade seja adequada para melhor aprendizado.

3.6.1 Função de Custo para RLMV

Tomando um conjunto de treino de tamanho m, a função custo para RLMV pode ser definida como

$$C(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 \tag{3.26}$$

e na forma matricial:

$$C(\theta) = \frac{1}{2m}(X\theta - y)^T(X\theta - y) \tag{3.27}$$

Quanto menor for o valor de C, melhor. Para isso há o método do Gradiente Descendente.

3.7 Gradient Descent

O método do gradiente descendente procura minimizar localmente uma função por meio de pequenos passos na direção de maior variação possível (direção do gradiente) . Ou seja, partindo de um vetor inicial , vamos atualizando-o na direção de sua derivada segundo uma taxa de aprendizado α

$$\theta_j := \theta_j - \alpha\frac{\partial(J)}{\partial\theta_j} \tag{3.28}$$

O método é extremamente útil ao tentarmos diminuir a função de custo (penalidade) no aprendizado das redes neurais, como a seguir:

$$\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x^{(i)} \tag{3.29}$$

e na forma matricial:

$$\theta_j := \theta_j - \frac{\alpha}{m} X^T (X\theta - y) \tag{3.30}$$

sendo X a matriz com os exemplos de treinamento e y os *labels*.

3.8 Overfitting

Overfitting ocorre quando o algoritmo treinado não tem a capacidade de generalização , sendo sua capacidade preditiva satisfatória somente no conjunto de treino, ou seja, o modelo ficou complexo demais a ponto de predizer os ruídos do conjunto de treino. Também é comum quando a quantidade de dados não é suficiente . Para evitar tal situação , foi usada a regularização L2 no presente trabalho.

3.8.0.1 Função Custo Regularizada

Para que não haja *overfit* é acrescentado um termo adicional no cálculo do erro de predição, de modo que há menor complexidade no treinamento dado o conjunto de treino disponível.

$$C(\theta) = \frac{1}{2m} \left[\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2 \right] \tag{3.31}$$

O parâmetro de regularização λ determina o quanto o custo dos parâmetros θ serão penalizados. No presente trabalho $\theta = 0.01$ mostrou-se por erro e tentativa, satisfatório.

3.9 Redes Neurais

Uma Rede Neural é uma estrutura matemática composta por nós e arestas(ligações), similar a um grafo. A informação entra na camada de **input**, se propaga pelas camadas intermediárias, em que toda a complexidade do problema é abstraída até finalmente se gerar a saída. A informação se propaga tanto para frente (*Forward Propagation*), como na direção reversa (*Backward Propagation*) .

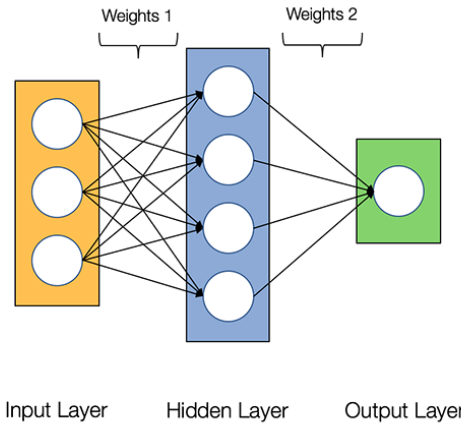


FIGURA 3.6 – Exemplo de uma Rede Neural, retirada de Aurélien-2018.

3.9.1 Estruturas

São elementos fundamentais: entradas, saídas, parâmetros e funções de ativação. A ideia é simples, basta se realizar o produto interno entre o vetor *input* de uma camada e o vetor de pesos, de modo que tal resultado é a entrada de uma função de ativação para que se gere uma saída e então continua-se o método de *Forward Propagation*

Algumas funções de ativação são :

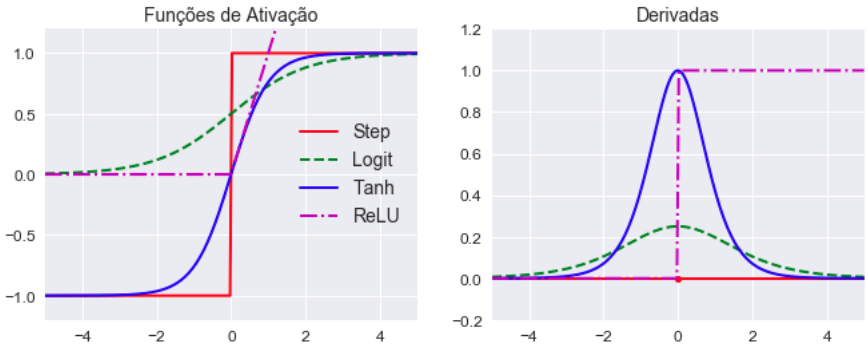


FIGURA 3.7 – Funções de Ativação e suas derivadas, plotadas em Python

O produto interno é :

$$\begin{bmatrix} x_0 \\ x_1 \\ ... \\ x_n \end{bmatrix} \rightarrow \begin{bmatrix} \text{Unidades de} \\ \text{Ativação} \end{bmatrix} \rightarrow h_\theta(x)$$

O valor de cada unidade de ativação é obtido da seguinte forma

$$a_i^{(j)} = f(\Theta_{i0}^{(k-1)}x_0 + \Theta_{i1}^{(k-1)}x_1 + ... + \Theta_{in}^{(k-1)}x_n) \tag{3.32}$$

$$a_i^{(j)} = f(z_i^{(j)}) \tag{3.33}$$

com

$$z_i^{(j)} = \Theta_{i0}^{(k-1)}x_0 + \Theta_{i1}^{(k-1)}x_1 + ... + \Theta_{in}^{(k-1)}x_n \tag{3.34}$$

e ainda,

$$z^{(k)} \begin{bmatrix} z_1^{(k)} \\ z_2^{(k)} \\ ... \\ z_n^{(k)} \end{bmatrix}$$

$$z^{(j)} = \Theta^{(k-1)}a^{(k-1)} \tag{3.35}$$

3.9.2 Função Custo

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\left[y_k^{(i)}\log\left((h_\Theta(x^{(i)})_k\right) + (1-y_k^{(i)})\log\left(1-(h_\Theta(x^{(i)})_k\right)\right] + \\ +\lambda\frac{1}{2m}\sum_{l=1}^{L}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}(\Theta_{j,i}^{(l)})^2 \tag{3.36}$$

Para redes neurais a função de custo é uma forma mais geral da utilizada na regressão logística, sendo :

- L: número total de camadas na rede;
- s_l: número de unidades de ativação na camada l (sem contar a unidade de *Bias*); e
- K: número de unidades de saída[1]

[1]Também chamado de classes, utilizado em problemas de classificação.

3.9.3 Foward Propagation

Definida a estrutura básica de uma NN e os cálculos envolvidos, pode-se definir o algoritmo de *Foward Propagation*. Ele consiste em: para uma entrada composta pela matriz X, calcular as saídas de todas as camadas ($a^{(l)}$ e $z^{(l)}$), incluindo a saída final da NN.

Algorithm 1: Forward Propagation

for $l = 1, ..., L$ **do**
  $z^{(l)} = \Theta^{(l-1)} a^{(l-1)}$
  $a^{(l)} = g(z^{(l)})$
end
return $a^{(L)}$

3.9.4 Backpropagation

- $\Delta^{(l)}$: acumulador da derivada parcial
- $\delta^{(L)}$: erro entre o valor observado e a saída da rede. Para as camadas anteriores, o valor de δ é calculado a partir da derivada da função de ativação:

$$\delta^{(l)} = \left(\left(\Theta^{(l)}\right)^T \delta^{(l+1)} \right) g'\left(a^{(l)}\right) \tag{3.37}$$

- $D_{ij}^{(l)}$: Derivada parcial da função custo em relação a $\Theta_{ij}^{(l)}$

Algorithm 2: Backpropagation

Set $\Delta = 0$
for $i = 1, ..., m$ **do**
  set $a^{(1)} = x^{(i)}$; aplicar foward propagation para encontrar $a^{(l)}$
  usando $y^{(i)}$, encontrar $\delta^{(L)}$
  calcular $\delta^{(L-1)}$,$\delta^{(L-2)}$,...,$\delta^{(2)}$
  $\Delta^{(l)} := \Delta^{(l)} + \delta^{(l+1)} \left(a^{(l)}\right)^T$
end
$D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)}$ para $j \neq 0$
$D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)}$ para $j = 0$
return D

3.9.5 Método Gradiente

Algorithm 3: Gradient Descent

Inicializar Θ com valores aleatórios próximos de 0
while *Um valor mínimo da função custo não é atingido* **do**
    Forward Propagation()
    Backward Propagation()
    for $l = 1, ..., L$ **do**
        $\Theta^{(l)} := \Theta^{(l)} - \alpha D^{(l)}$
    end
end
return Θ

3.9.6 Random Forests

O Algoritmo *Random Forest* mostrou-se bastante promissor para a modelagem dos coeficientes aerodinâmicos.

Algorithm 4: Random Forest

Sample m data sets $D_1, D_2, ..., D_m$ from D with replacement

for $i = 1, ..., m$ **do**

  train a full depth decision tree $h_i()$ (max-depth = ∞) considering $k <= d$, where d is the dimension of D , features and without replacement to split the decision tree

end

$h(x) = \frac{\sum_{l=1}^{L} h_i(x)}{m}$

return $h(x)$

Da descrição do algoritmo dada anteriormente, vemos que há apenas dois hiper parâmetros, k e m. Segundo (Philipp Probs), um valor satisfatório para k é $k = \sqrt{d}$, já m pode ser suficientemente grande, desde que não afete a velocidade do treinamento e nem comprometa os recursos de memória disponíveis. Um valor grande de m não acarreta problemas de $overfitting$ nem compromete o erro durante treinamento. Outra vantagem do algoritmo, especialmente neste trabalho onde a quantidade de dados não é elevada, é que não é necessário dividir o data set em conjunto de treino, validação e teste, dado que se trata de um algoritmo *bagging*, ou seja, pode se usar todo o conjunto de dados disponível para treinamento. Também é possível avaliar a importância de cada *feature* com relação à predição dos *labels* quando o algoritmo realiza o corte das *features*. Neste trabalho, como será verificado adiante, tal método confirmou que a varredura em $alfa$ está fortemente correlacionada com as saídas C_D, C_L e C_M e a varredura em β está fortemente correlacionada com C_N, C_Y e C_1 .

Uma característica importante da *Random Forest* é da diminuição da variância durante a predição, e tal característica é bastante pertinente com relação à predição dos coeficientes aerodinâmicos, devido à natureza altamente dispersiva dos mesmos. Como a predição é feita em várias árvores de decisão ,obtém-se uma precisão maior e variância média menor. Principalmente em problemas de regressão, tais incertezas são difíceis de obter por outros métodos .Além disso, a abordagem *out of the bag* fornece uma estimativa não enviesada . Matematicamente, para cada ponto $(x_i, y_i) \notin D$, onde D é o conjunto de treino, seja $S_i = k|(x_i, y_i) \notin D_k$, ou seja , S_i é o conjunto de todos os *datasets* D_k que não contém (x_k, y_k). Sendo o algoritmo de regressão em torno de todos esses *datasets* , dado por $H_i(x) = \frac{\sum_{l=1}^{L} h_i(x)}{|S_i|}$, o erro total é simplesmente a média dos erro de tais algoritmos, sendo l uma função que calcula o erro de predição:

$$e(x) = \frac{\sum_{(x_i,y_i)\in D} l(H_i, y_i)}{n}$$

Ou seja, a variância é distribuída entre várias árvores de decisão e ,no cálculo da média ,obtemos uma menor variância total.

4 Data Mining

O processo de mineração de dados (em inglês *data mining*) consiste na obtenção, limpeza e preparação de dados, encontrando comportamentos e associações relevantes.

As principais abordagens utilizadas para a limpeza e preparação dos dados foram: Visualização dos dados e Análise Estatística Básica. Na visualização de dados, foi aplicado desde o conceito de OCR para uma primeira manipulação dos dados à visualização de *outliers* e correlações na amostra. A análise estatística básica permitiu a medida da tendência central com os conceitos de média e mediana, além das medidas de dispersão dos dados com os conceitos de quartis e desvio padrão.

4.1 Amostra de Estudo

Neste trabalho, selecionamos as condições da aeronave em decolagem, cruzeiro e pouso. Foram estudados os ensaios, à pressão dinâmica de $270 mm H_2O$ constante, com variação do ângulo de:

- profundor de $-20°$ a $+20°$, com passo de $10°$
- leme de $0°$, $+10°$, $+20°$ e $+25°$
- ailerons de $+17°/-15°$, $+11,33°/-10°$ e $+5,66°/-5°$
- ataque de $-10°$ a $+30°$, com passo de $2°$ e na região de estol o passo foi reduzido para $1°$
- guinada de $-30°$ a $+30°$, com passo de $5°$

4.2 *Open CV*

Os dados do ensaio em túnel de vento do modelo do TUCANO foram fornecidos em tabelas não-editáveis em formato *pdf*. Pela grande massa de dados contida nos arquivos,

seria necessária uma grande disposição de tempo para digitar todos os dados. A alternativa encontrada foi utilizar a ferramenta *Tesseract* ,que dispões de um método OCR (*Optical Recognition Character*) para reconhecimento de texto em imagem. Então foram tirados diversos *screenshots* das tabelas dos ensaios para extração dos dados. Inicialmente a extração dos dados mostrou-se imprecisa devida à má resolução das imagens. Infelizmente, a função *ocr()* não é capaz de fornecer resultados precisos sempre. Existem muitos erros de leitura que atrapalham a análise de dados, como números interpretados como letras e identificação errada de números. Tudo isso gera *outliers* e prejudica o resultado final da modelagem.

A solução adotada foi pré-processar as imagens *png* tiradas do arquivo *pdf* com a ferramenta Open CV, para que a leitura fosse feita em uma imagem com melhor resolução. Basicamente é realizado o seguinte procedimento:

1. Conversão de imagem para *gray scale*
2. Aumentar o brilho da imagem
3. Executar transformações morfológicas para melhorar a qualidade do texto

Como a imagem está embaraçada, podemos usar usar o seguinte *kernel* para aumentar o brilho da imagem :

$$\omega = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{4.1}$$

Tal *kernel* é conhecido dentro da teoria de processamento de imagens para ajuste de brilho e reconhecimento de arestas em imagens. Após esse processamento ,a imagem ainda apresentou buracos no texto ,portanto ,foi feita uma transformação morfológica de dilatação retangular para fechar os buracos e tornar o texto mais nítido.

Considerando *image* como uma das entradas dos ensaio, o seguinte *snippet* mostra como foi realizado tal processamento.

```
1
2 import cv2
3 import sys
```

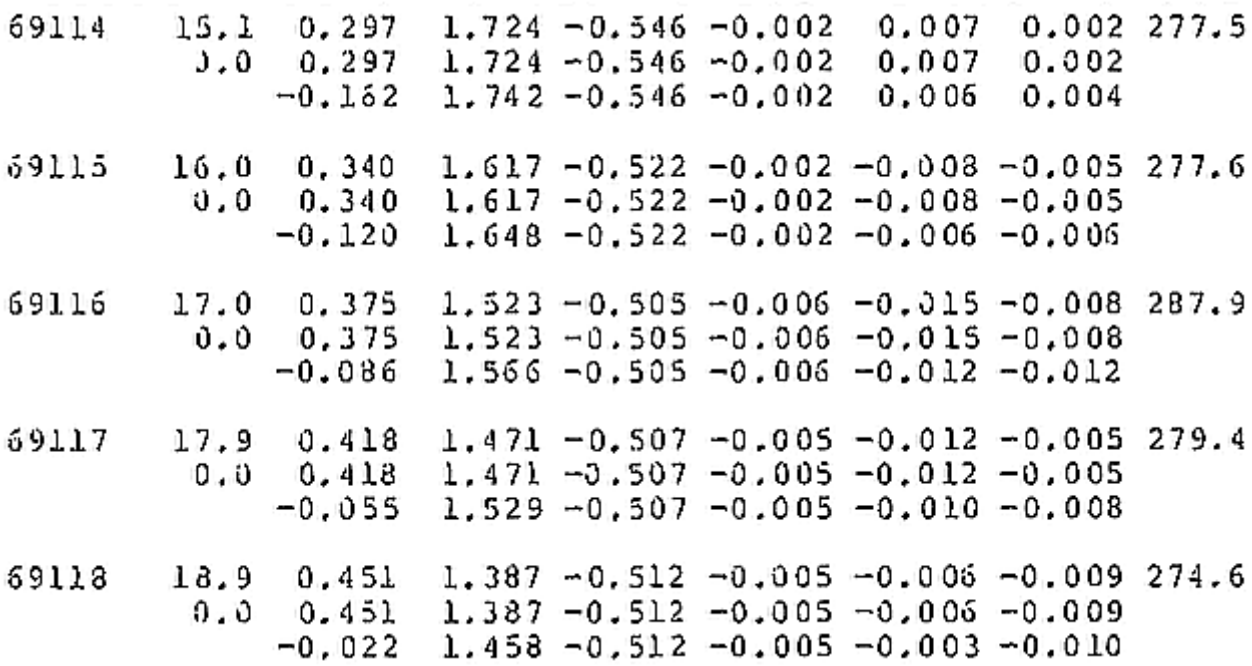
```
4  import pytesseract
5
6  import numpy as np
7
8  image = cv2.imread('image.png')
9  gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
10 sharpen_kernel = np.array([[-1,-1,-1], [-1,9,-1], [-1,-1,-1]])
11 sharpen = cv2.filter2D(gray, -1, sharpen_kernel)
12
13 cv2.imwrite('sharpen.png', sharpen)
14 sharpen = 255 - sharpen
15 kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (2,2))
16 dilate = cv2.dilate(sharpen, kernel, iterations=1)
17
18 result = 255 - dilate
19 cv2.imwrite('result.png', result)
20
21 text = pytesseract.image_to_string(result)
22 toCsv(text)
```

Listing 4.1 – Processamento das imagens dos ensaios de túnel de vento

Sendo as saídas mostradas a seguir :

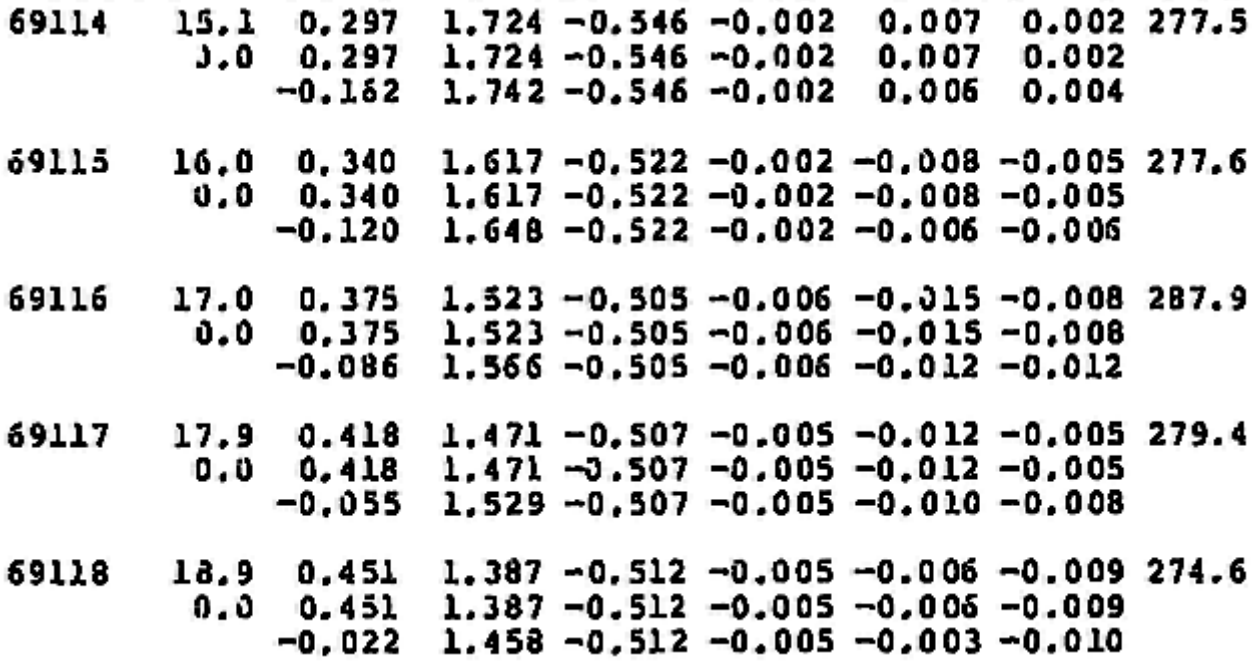
```
69114    15.1   0.297   1.724 -0.546 -0.002   0.007   0.002 277.5
          0.0   0.297   1.724 -0.546 -0.002   0.007   0.002
               -0.162   1.742 -0.546 -0.002   0.006   0.004

69115    16.0   0.340   1.617 -0.522 -0.002 -0.008 -0.005 277.6
          0.0   0.340   1.617 -0.522 -0.002 -0.008 -0.005
               -0.120   1.648 -0.522 -0.002 -0.006 -0.006

69116    17.0   0.375   1.523 -0.505 -0.006 -0.015 -0.008 287.9
          0.0   0.375   1.523 -0.505 -0.006 -0.015 -0.008
               -0.086   1.566 -0.505 -0.006 -0.012 -0.012

69117    17.9   0.418   1.471 -0.507 -0.005 -0.012 -0.005 279.4
          0.0   0.418   1.471 -0.507 -0.005 -0.012 -0.005
               -0.055   1.529 -0.507 -0.005 -0.010 -0.008

69118    18.9   0.451   1.387 -0.512 -0.005 -0.006 -0.009 274.6
          0.0   0.451   1.387 -0.512 -0.005 -0.006 -0.009
               -0.022   1.458 -0.512 -0.005 -0.003 -0.010
```

FIGURA 4.1 – Captura de tela do arquivo pdf original dos dados do túnel de vento



```
69114   15.1   0.297   1.724 -0.546 -0.002  0.007  0.002 277.5
         0.0   0.297   1.724 -0.546 -0.002  0.007  0.002
              -0.162   1.742 -0.546 -0.002  0.006  0.004

69115   16.0   0.340   1.617 -0.522 -0.002 -0.008 -0.005 277.6
         0.0   0.340   1.617 -0.522 -0.002 -0.008 -0.005
              -0.120   1.648 -0.522 -0.002 -0.006 -0.006

69116   17.0   0.375   1.523 -0.505 -0.006 -0.015 -0.008 287.9
         0.0   0.375   1.523 -0.505 -0.006 -0.015 -0.008
              -0.086   1.566 -0.505 -0.006 -0.012 -0.012

69117   17.9   0.418   1.471 -0.507 -0.005 -0.012 -0.005 279.4
         0.0   0.418   1.471 -0.507 -0.005 -0.012 -0.005
              -0.055   1.529 -0.507 -0.005 -0.010 -0.008

69118   18.9   0.451   1.387 -0.512 -0.005 -0.006 -0.009 274.6
         0.0   0.451   1.387 -0.512 -0.005 -0.006 -0.009
              -0.022   1.458 -0.512 -0.005 -0.003 -0.010
```

FIGURA 4.2 – Imagem após convolução e realizando uma operação de subtração nos pixels



```
69114   15.1   0.297   1.724 -0.546 -0.002  0.007  0.002 277.5
         0.0   0.297   1.724 -0.546 -0.002  0.007  0.002
              -0.162   1.742 -0.546 -0.002  0.006  0.004

69115   16.0   0.340   1.617 -0.522 -0.002 -0.008 -0.005 277.6
         0.0   0.340   1.617 -0.522 -0.002 -0.008 -0.005
              -0.120   1.648 -0.522 -0.002 -0.006 -0.006

69116   17.0   0.375   1.523 -0.505 -0.006 -0.015 -0.008 287.9
         0.0   0.375   1.523 -0.505 -0.006 -0.015 -0.008
              -0.086   1.566 -0.505 -0.006 -0.012 -0.012

69117   17.9   0.418   1.471 -0.507 -0.005 -0.012 -0.005 279.4
         0.0   0.418   1.471 -0.507 -0.005 -0.012 -0.005
              -0.055   1.529 -0.507 -0.005 -0.010 -0.008

69118   18.9   0.451   1.387 -0.512 -0.005 -0.006 -0.009 274.6
         0.0   0.451   1.387 -0.512 -0.005 -0.006 -0.009
              -0.022   1.458 -0.512 -0.005 -0.003 -0.010
```

FIGURA 4.3 – Imagem final

Com os dados extraídos em formato *txt* foi realizado mais um processamento, como formatação, conversão dos tipos de dados para que finalmente os dados estivessem aptos a serem analisados no ambiente *Python*.

4.2.1 Data Cleaning

Com os dados prontos, estes foram armazenados como arquivos *csv* e armazenados em *Git Hub* próprio, para que pudessem ser transferidos para o ambiente *Python* . Para o restante do trabalho, os dados a serem trabalhados estão separados e nomeados da seguinte forma, considerando as diferentes configurações do ensaio de túnel de vento, e também a quantidade de dados por variável, definidas no capítulo 2:

1. Ensaios com varredura em α
   (a) decolagemAlfa : 2189
   (b) recolhidoAlfa : 2306
   (c) aterragemAlfa : 2083
2. Ensaios com varredura em β
   (a) decolagemBeta : 1232
   (b) recolhidoBeta : 1133
   (c) aterragemBeta : 1188

Obtendo os dados armazenados em *Git Hub* e convertendo-os em *csv*

```
1
2 aterragem_alfa = pd.read_csv('https://raw.githubusercontent.com/...
    snowfit/Tese-de-Gradua-o---ITA-2019/master/dados/dataset0.csv',...
    error_bad_lines=False)
3 aterragem_beta = pd.read_csv('https://raw.githubusercontent.com/...
    snowfit/Tese-de-Gradua-o---ITA-2019/master/dados/dataset1.csv',...
    error_bad_lines=False)
4 decolagem_alfa = pd.read_csv('https://raw.githubusercontent.com/...
    snowfit/Tese-de-Gradua-o---ITA-2019/master/dados/dataset2.csv',...
    error_bad_lines=False)
5 decolagem_beta = pd.read_csv('https://raw.githubusercontent.com/...
    snowfit/Tese-de-Gradua-o---ITA-2019/master/dados/dataset3.csv',...
    error_bad_lines=False)
6 recolhido_alfa = pd.read_csv('https://raw.githubusercontent.com/...
    snowfit/Tese-de-Gradua-o---ITA-2019/master/dados/dataset4.csv',...
    error_bad_lines=False)
7 recolhido_beta = pd.read_csv('https://raw.githubusercontent.com/...
    snowfit/Tese-de-Gradua-o---ITA-2019/master/dados/dataset5.csv',...
    error_bad_lines=False)
```

Em teoria, as *Random Forests* poderiam trabalhar com dados ausentes e categóricos. No entanto, sua implementação no *TensorFlow* não lida com isso. Para preparar dados para *Random Forests* no pacote *Python*, é necessário que:

1. Não exista valores ausentes nos dados
2. Dados categóricos sejam convertidos em numéricos

O pré-processamento de dados para redes neurais requer o preenchimento de valores ausentes e a conversão de dados categóricos em numéricos. Além disso, é necessário o dimensionamento de *features*. No caso de dados com dimensões muito diferentes, haverá problemas com o treinamento do modelo. Se você não dimensionar *features* nos mesmos intervalos, aquelas com valores maiores serão tratadas como mais importantes no treinamento, o que não é desejado. Além disso, os valores dos gradientes podem explodir e os neurônios podem saturar, o que tornará impossível o treinamento da Rede. Portanto, para o treinamento da Rede Neural, você precisa fazer o seguinte pré-processamento:

1. Preencher valores ausentes
2. Converter dados categóricos em numéricos
3. Dimensionar os dados no mesmo intervalo (ou pelo menos semelhante)

É necessário lembrar que todo o pré-processamento usado para preparar dados de treinamento deve ser usado na produção. Para a Rede, você tem mais etapas de pré-processamento, portanto, mais etapas a serem implementadas no sistema de produção.

Tais etapas foram realizadas como a seguir :

```
1
2 def drop_na(df):
3   return df.dropna()
4
5 def drop_outliers(df,features):
6
7   Q = df[features]
8
9   Q1 = Q.quantile(0.25)
10  Q3 = Q.quantile(0.75)
11  IQR = Q3 - Q1
12  new_df = df[¬((Q < (Q1 - 3 * IQR)) |(Q > (Q3 + 3 * IQR))).any(...
     axis=1)]
13  return new_df
14
15
16 features = ['CD','CL','CM','CY','C1','CN','alfa','beta']
17 datas = [decolagemAlfa,decolagemBeta,recolhidoAlfa,recolhidoBeta,...
     aterragemAlfa,aterragemBeta]
18
19 decolagemAlfa = decolagemAlfa.dropna()
20 decolagemBeta = decolagemBeta.dropna()
21 recolhidoAlfa = recolhidoAlfa.dropna()
22 recolhidoBeta = recolhidoBeta.dropna()
23 aterragemAlfa = aterragemAlfa.dropna()
24 aterragemBeta = aterragemBeta.dropna()
25
26
27 decolagemAlfa = drop_outliers(decolagemAlfa,features)
28 decolagemBeta = drop_outliers(decolagemBeta,features)
29 recolhidoAlfa = drop_outliers(recolhidoAlfa,features)
30 recolhidoBeta = drop_outliers(recolhidoBeta,features)
31 aterragemAlfa = drop_outliers(aterragemAlfa,features)
32 aterragemBeta = drop_outliers(aterragemBeta,features)
```

Listing 4.2 – Tratamento de Outliers e valores Null

5 Análise dos Dados

Definida a amostra a ser estudada, é interessante verificar o comportamento dos coeficientes aeronáuticos para cada *input*. Os resultados se aproximam muito da literatura (STEVENS *et al.*, 2016).

5.1 Análise Gráfica

Então, da amostra de dados, foi feita uma varredura em cada entrada e verificando o comportamento das saídas, mantendo o valor de cada outra entrada zerado. Vale destacar que os seguintes gráficos estão em consonância com a teoria aerodinâmica. O coeficiente C_L mede o quão o aerofólio é capaz de gerar sustentação e tal coeficiente é linear para baixos valores de α e positivo para valores de α próximo a zero. C_D normalmente tem formato de parábola em função de α. Porém, para altos valores de α, o fluxo de ar não mais acompanha a superfície do aerofólio, de maneira que a distribuição de pressão muda e ocorre diminuição rápida da sustentação e o arrasto aumenta rapidamente. Nessas condições o aerofólio é dito estar em *stall*.

Decolagem Alfa

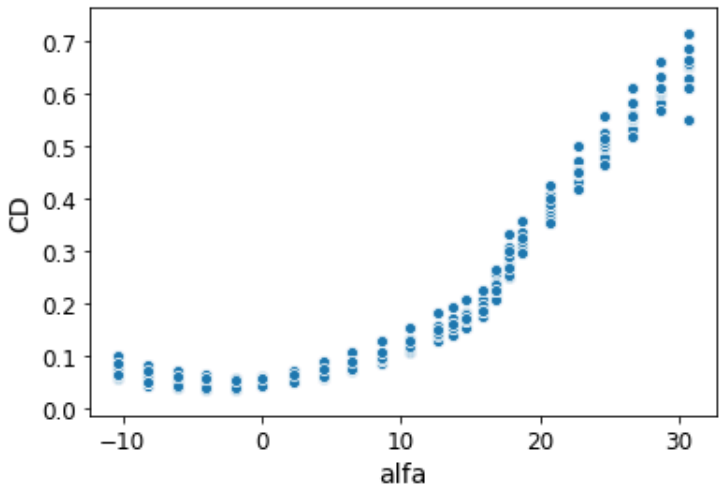


FIGURA 5.1 – Dispersão de C_D em função das entradas, com o eixo x indicando a variação da entrada e o eixo y o valor da variável de saída observada.

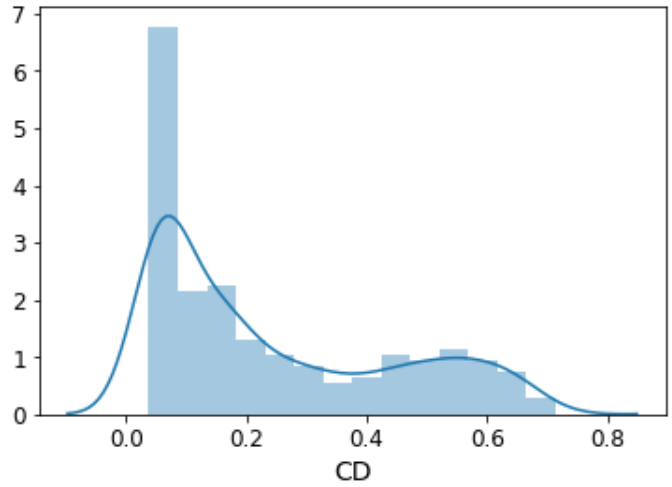


FIGURA 5.2 – Distribuição dos valores de C_D

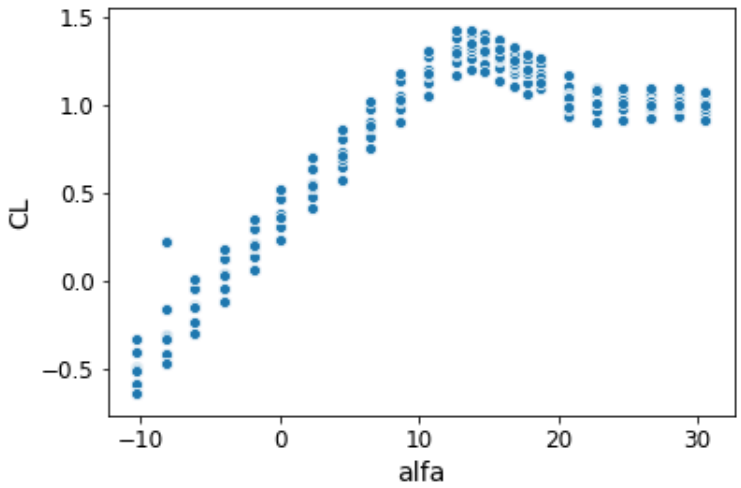


FIGURA 5.3 – Dispersão de C_L em função das entradas, com o eixo x indicando a variação da entrada e o eixo y o valor da variável de saída observada.

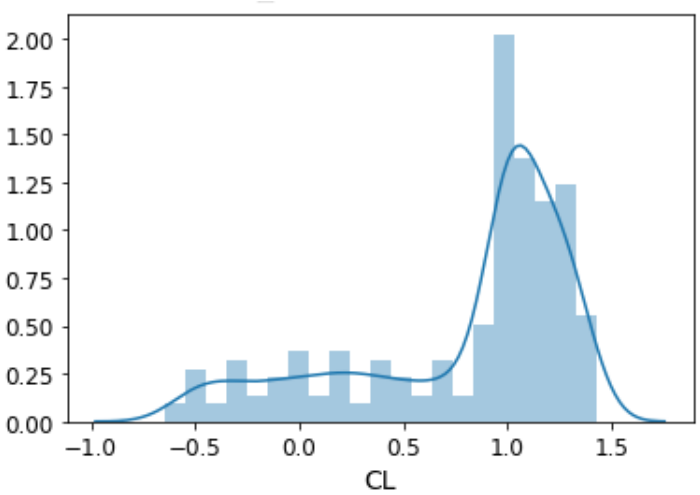


FIGURA 5.4 – Distribuição dos valores de C_L

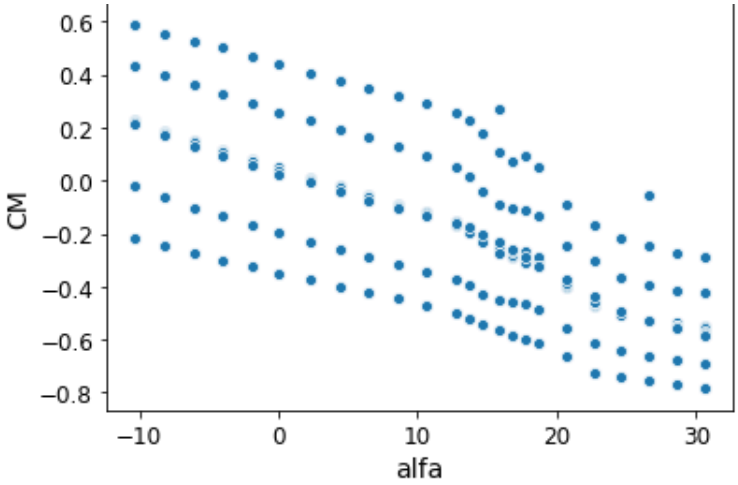


FIGURA 5.5 – Dispersão de C_M em função das entradas, com o eixo x indicando a variação da entrada e o eixo y o valor da variável de saída observada.

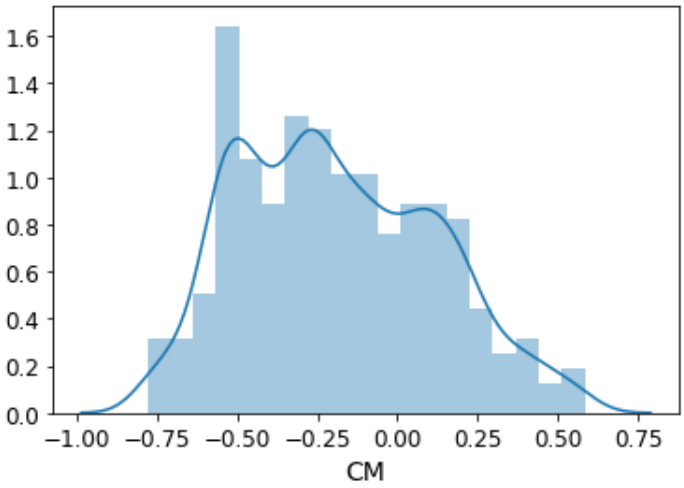


FIGURA 5.6 – Distribuição dos valores de C_M

Recolhido Alfa

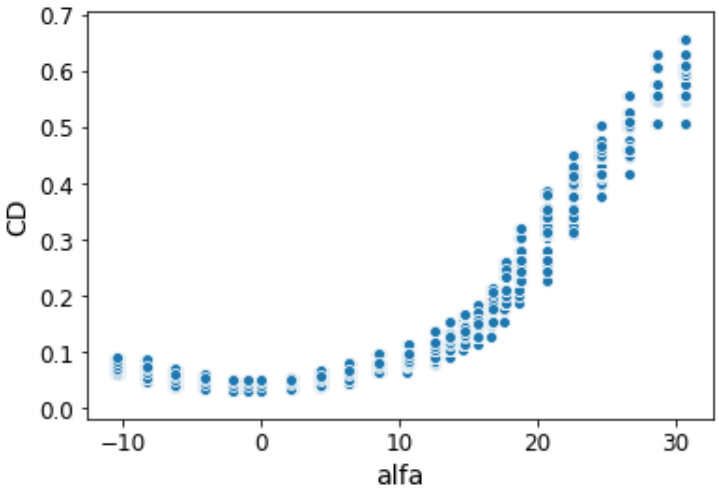


FIGURA 5.7 – Dispersão de C_D em função das entradas, com o eixo x indicando a variação da entrada e o eixo y o valor da variável de saída observada.

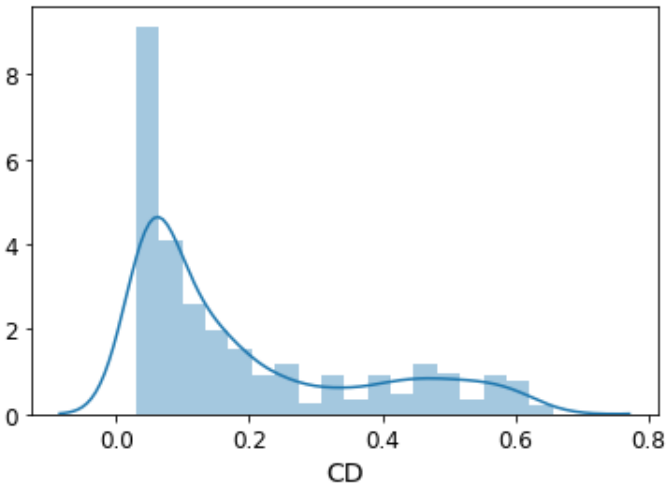


FIGURA 5.8 – Distribuição dos valores de C_D

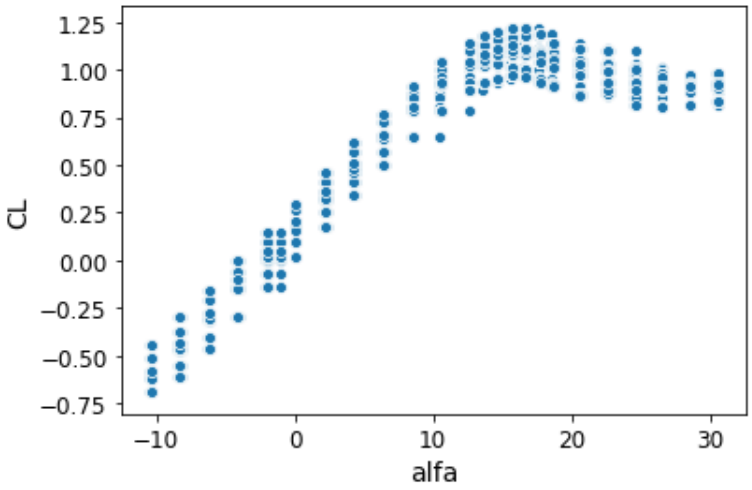


FIGURA 5.9 – Dispersão de C_L em função das entradas, com o eixo x indicando a variação da entrada e o eixo y o valor da variável de saída observada.

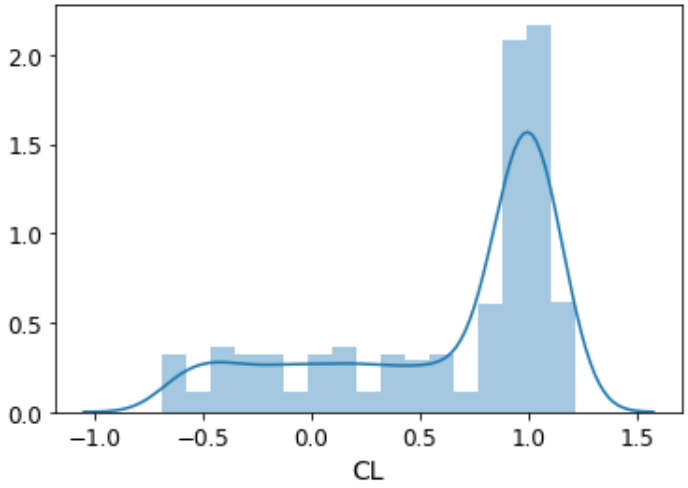


FIGURA 5.10 – Distribuição dos valores de C_L

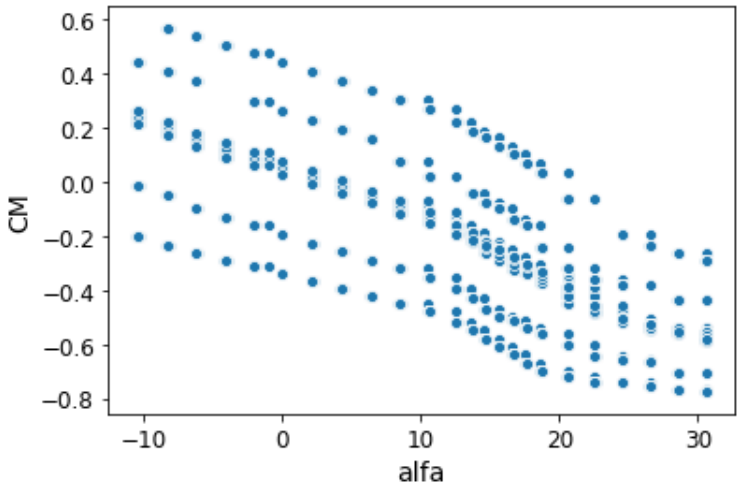


FIGURA 5.11 – Dispersão de C_M em função das entradas, com o eixo x indicando a variação da entrada e o eixo y o valor da variável de saída observada.

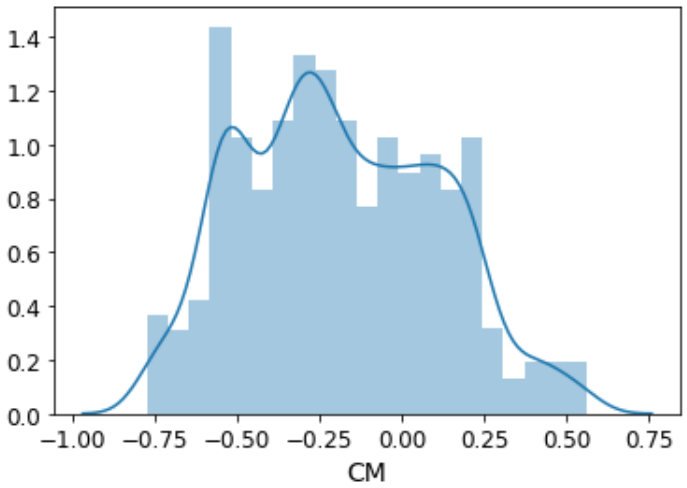


FIGURA 5.12 – Distribuição dos valores de C_M

Aterragem Alfa

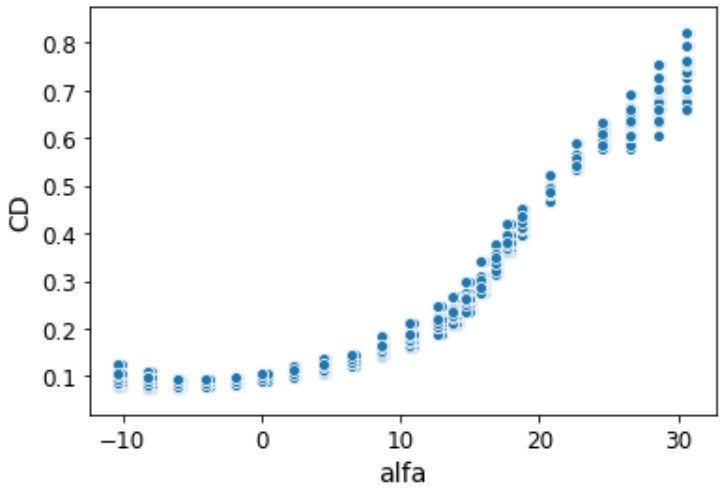


FIGURA 5.13 – Dispersão de C_D em função das entradas, com o eixo x indicando a variação da entrada e o eixo y o valor da variável de saída observada.

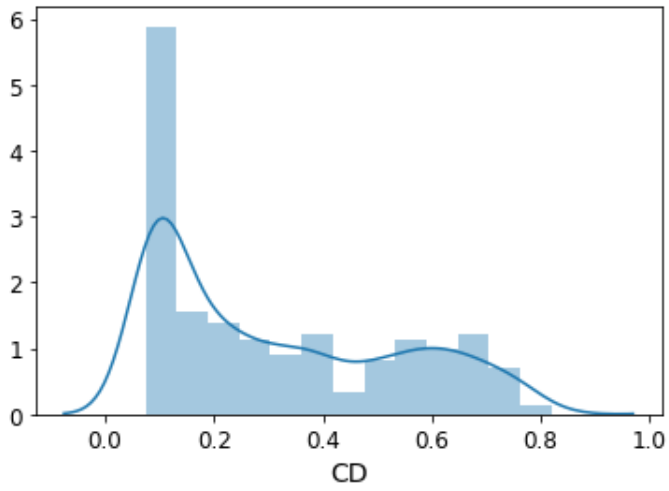


FIGURA 5.14 – Distribuição dos valores de C_D

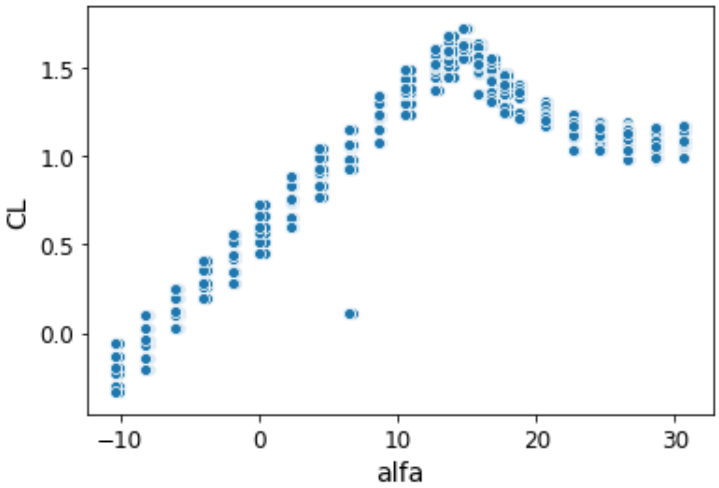


FIGURA 5.15 – Dispersão de C_L em função das entradas, com o eixo x indicando a variação da entrada e o eixo y o valor da variável de saída observada.

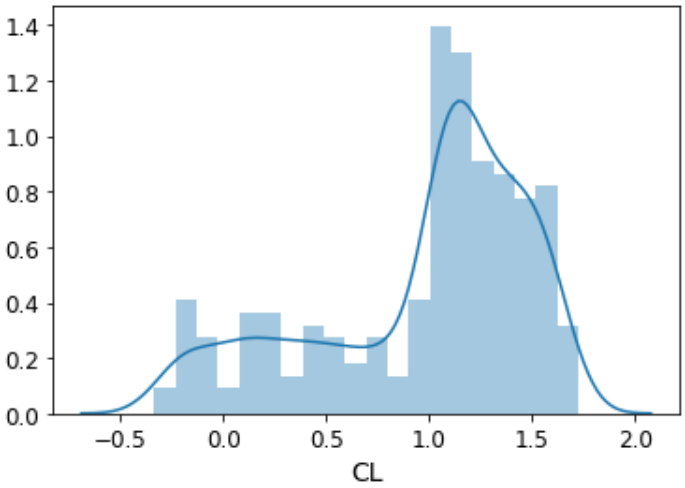


FIGURA 5.16 – Distribuição dos valores de C_L

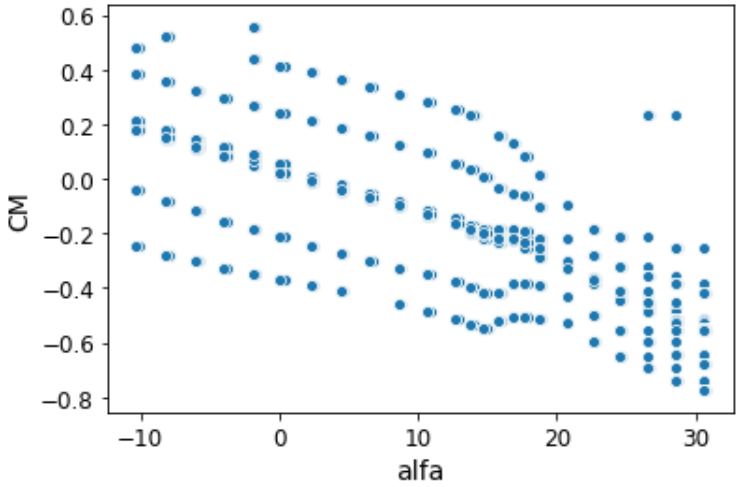


FIGURA 5.17 – Dispersão de C_M em função das entradas, com o eixo x indicando a variação da entrada e o eixo y o valor da variável de saída observada.

Ensaios em Beta

Para os resultados com varredura em Beta, foram plotados os gráficos considerando os três ensaios concatenados, por motivos que serão explicitados posteriormente no trabalho.

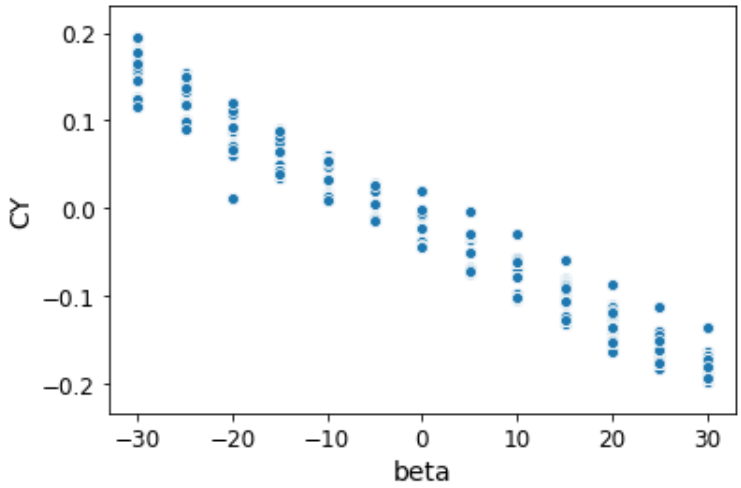


FIGURA 5.18 – Dispersão de C_Y em função das entradas, com o eixo x indicando a variação da entrada e o eixo y o valor da variável de saída observada.

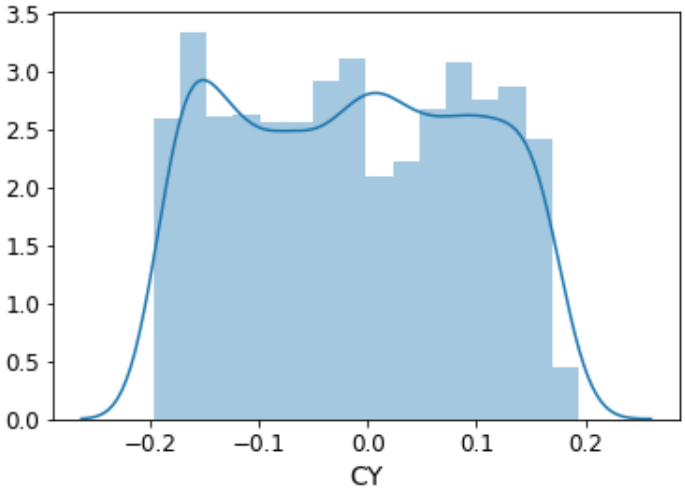


FIGURA 5.19 – Distribuição dos valores de C_Y

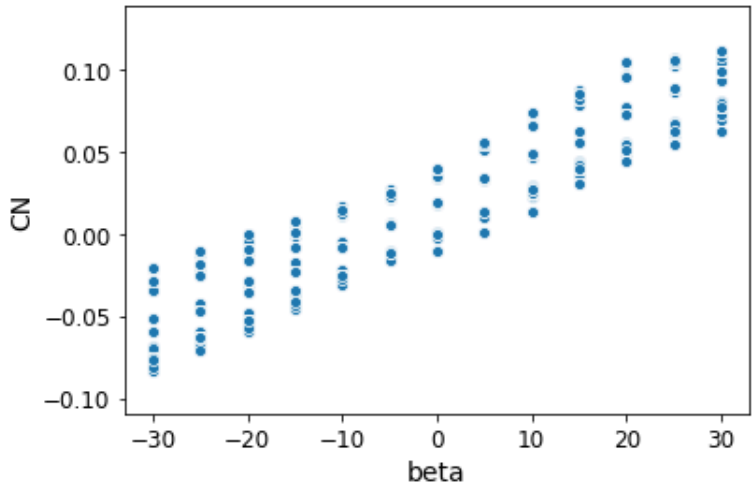


FIGURA 5.20 – Dispersão de C_N em função das entradas, com o eixo x indicando a variação da entrada e o eixo y o valor da variável de saída observada.

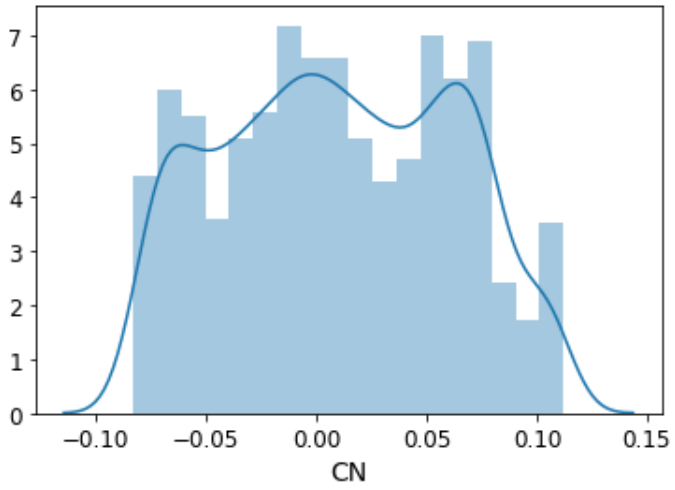


FIGURA 5.21 – Distribuição dos valores de C_N

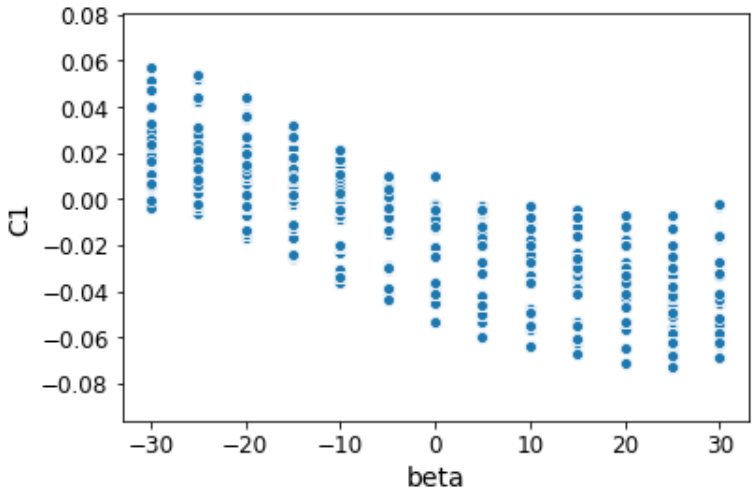


FIGURA 5.22 – Dispersão de C_1 em função das entradas, com o eixo x indicando a variação da entrada e o eixo y o valor da variável de saída observada.

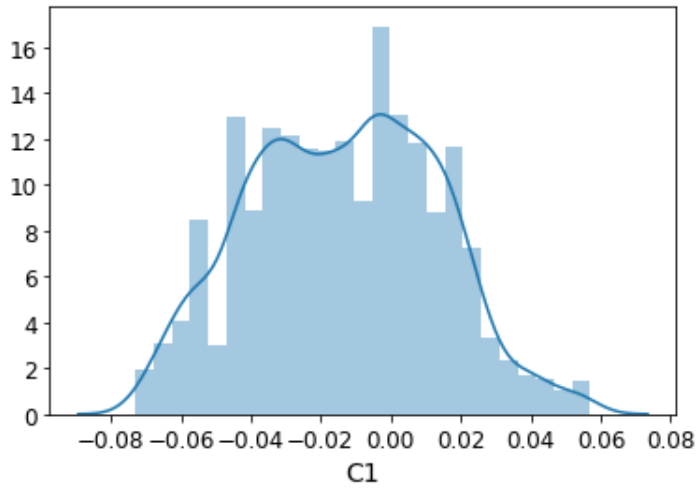


FIGURA 5.23 – Distribuição dos valores de C_1

Uma análise dos gráficos anteriores mostra a natureza dispersa dos coeficientes aerodinâmicos (alta variância). Portanto o algoritmo a ser treinado deverá ser capaz de diminuir a variância na predição. Tal natureza dos dados requer métodos sofisticados para a modelagem de suas não linearidades. A Rede Neural e a *Random Forest* fornecem uma modelagem com baixo erro quadrático médio ,como será analisado posteriormente.

5.2 Análise Estatística

Os ensaios no túnel de vento foram realizados em três configurações, com flape variando em 0,12 e 35 graus, com varredura nos ângulos de ataque (α) e guinada (β) , cada um influenciando nos coeficientes de Força e Momentos, respectivamente. As distribuições

dos valores de Força foram bastante diferentes, o que condiz com a teoria aerodinâmica, dado que o flape altera basicamente a sustentação, já para os Momentos não se apresentou diferença estatística significativa. Uma análise das médias e desvios padrões é fornecida a seguir:

Variável	Média	Desvio Padrão
CD	0.241209	0.199142
CL	0.803886	0.528908
CM	-0.186881	0.300680

TABELA 5.1 – Decolagem Alfa

Variável	Média	Desvio Padrão
CD	0.191176	0.174830
CL	0.620945	0.542890
CM	-0.185682	0.297079

TABELA 5.2 – Recolhido Alfa

Variável	Média	Desvio Padrão
CD	0.300074	0.217034
CL	0.96220	0.53457
CM	-0.158204	0.271533

TABELA 5.3 – Aterragem Alfa

Variável	Média	Desvio Padrão
CY	-0.011766	0.106978
C1	-0.012340	0.025554
CN	0.009496	0.049983

TABELA 5.4 – Decolagem Beta

Variável	Média	Desvio Padrão
CY	-0.010239	0.108365
C1	-0.012279	0.026330
CN	0.009357	0.050303

TABELA 5.5 – Recolhido Beta

Variável	Média	Desvio Padrão
CY	-0.00978	0.10601
C1	-0.012795	0.029630
CN	0.008288	0.053152

TABELA 5.6 – Aterragem Beta

Analisando as tabelas anteriores, verificou-se que os ensaios com varredura em α apresentaram médias estatisticamente diferentes, já com relação aos ensaios com varredura em β verificou-se uma certa proximidade das métricas estatísticas. Para ratificar a hipótese de que as distribuições em α são diferentes e as em β são próximas, é calculada a probabilidade de que, dadas as hipóteses anteriores, é verificado os resultados fornecidos pelo ensaio.

```
1
2 features = ['CN','CY','C1']
3
4 for f in features:
5   d1 = decolagemBeta[f]
6   d2 = aterragemBeta[f]
7   for i in range(50):
8       # Gerando Amostras Permutadas
9       perm_sample_1, perm_sample_2 = permutation_sample(d1,d2)
10  # Computando a diferen a de m dias das sa das dos ...
     experimentos
11  empirical_diff_means = diff_of_means(d1,d2)
12  # Gerando 10.000 r plicas permutadas
13  perm_replicates = draw_perm_reps(d1,d2,diff_of_means, size...
     =10000)
14  print ('Empirical difference = ',empirical_diff_means)
15  # Calculando p-value
16  p = np.sum(perm_replicates ≥ empirical_diff_means) /len(...
     perm_replicates)
17  # Resultado
18  print('p-value =', p)
```

Listing 5.1 – Efetuando teste de hipótese com relação às saídas dos ensaios com varredura em β

Com relação aos ensaios com varredura em α, os *p-values* se aproximaram de zero, reforçando que devem ser tratados separadamente, já com relação aos ensaios com varredura em β, temos os seguintes valores.

1. decolagemBeta e recolhidoBeta
   (a) CN : 0.9397
   (b) CY : 0.7058
   (c) C1 : 0.9487
2. decolagemBeta e aterragemBeta
   (a) CN : 0.6012
   (b) CY : 0.913
   (c) C1 : 0.6294
3. recolhidoBeta e aterragemBeta
   (a) CN : 0.5419

(b) CY : 0.6258

(c) C1 : 0.6696

Estatisticamente, pode-se considerar os três ensaios como gerados por uma mesma distribuição. Portando gerou-se um novo *Dataset* :

```
1
2 momentsDataset = pd.concat([decolagemBeta,recolhidoBeta,...
      aterragemBeta])
```

Listing 5.2 – Concatenando dados

5.3 Correlações

Agora, feita uma primeira análise gráfica, é possível tentar observar as correlações existentes entre os dados.

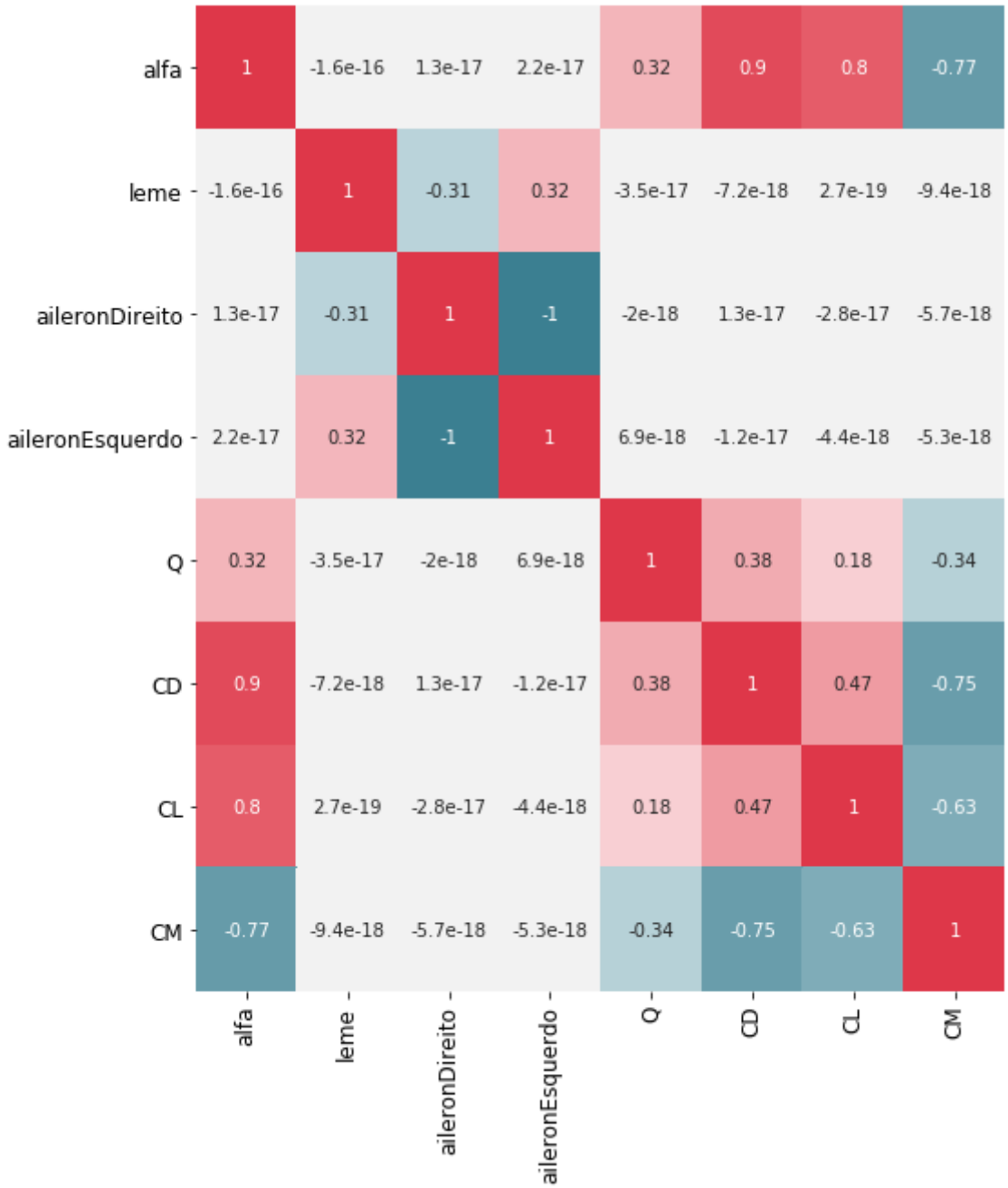


FIGURA 5.24 – Heat Map no Data Set Decolagem Alfa

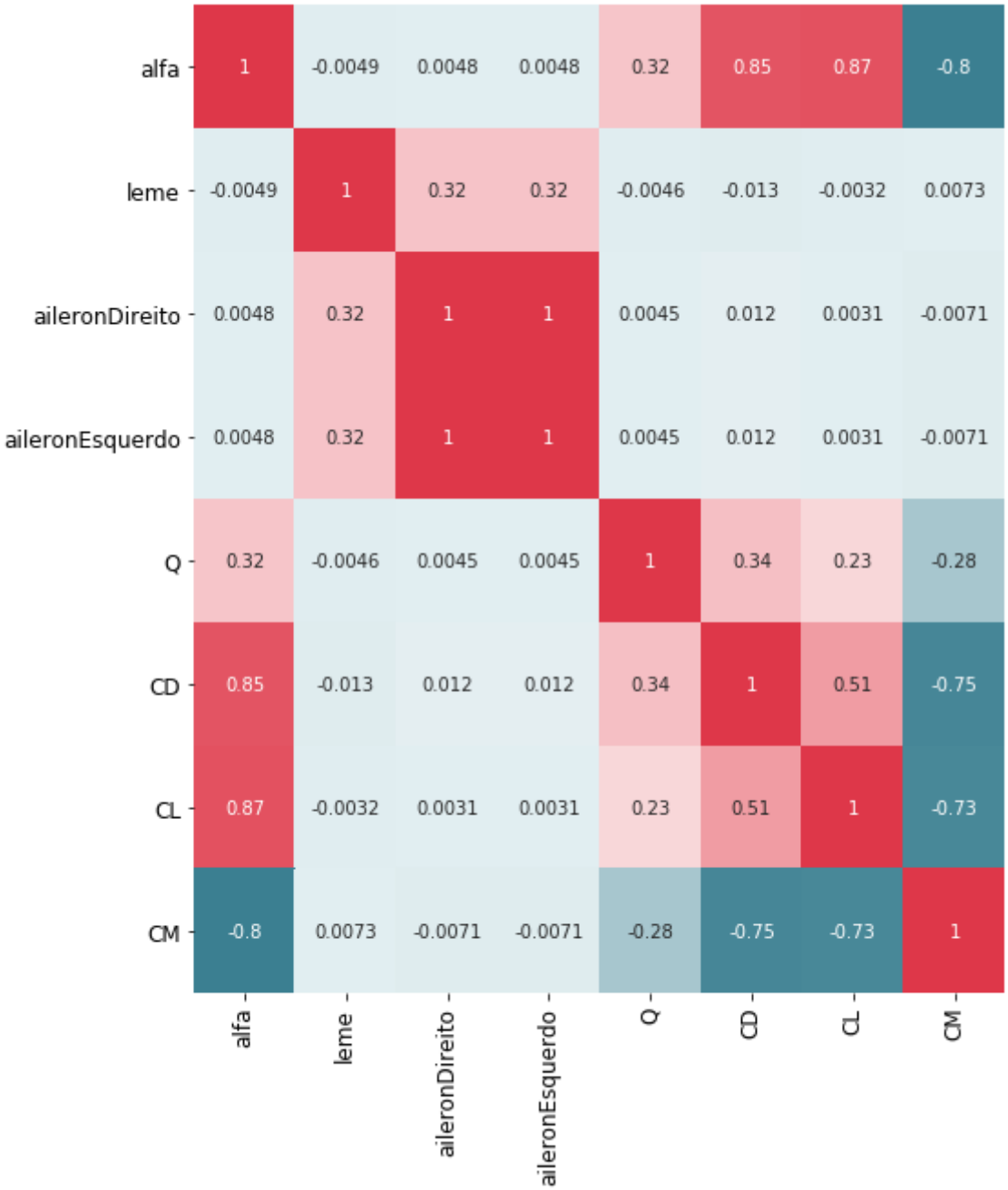


FIGURA 5.25 – Heat Map no Data Set Recolhido Alfa

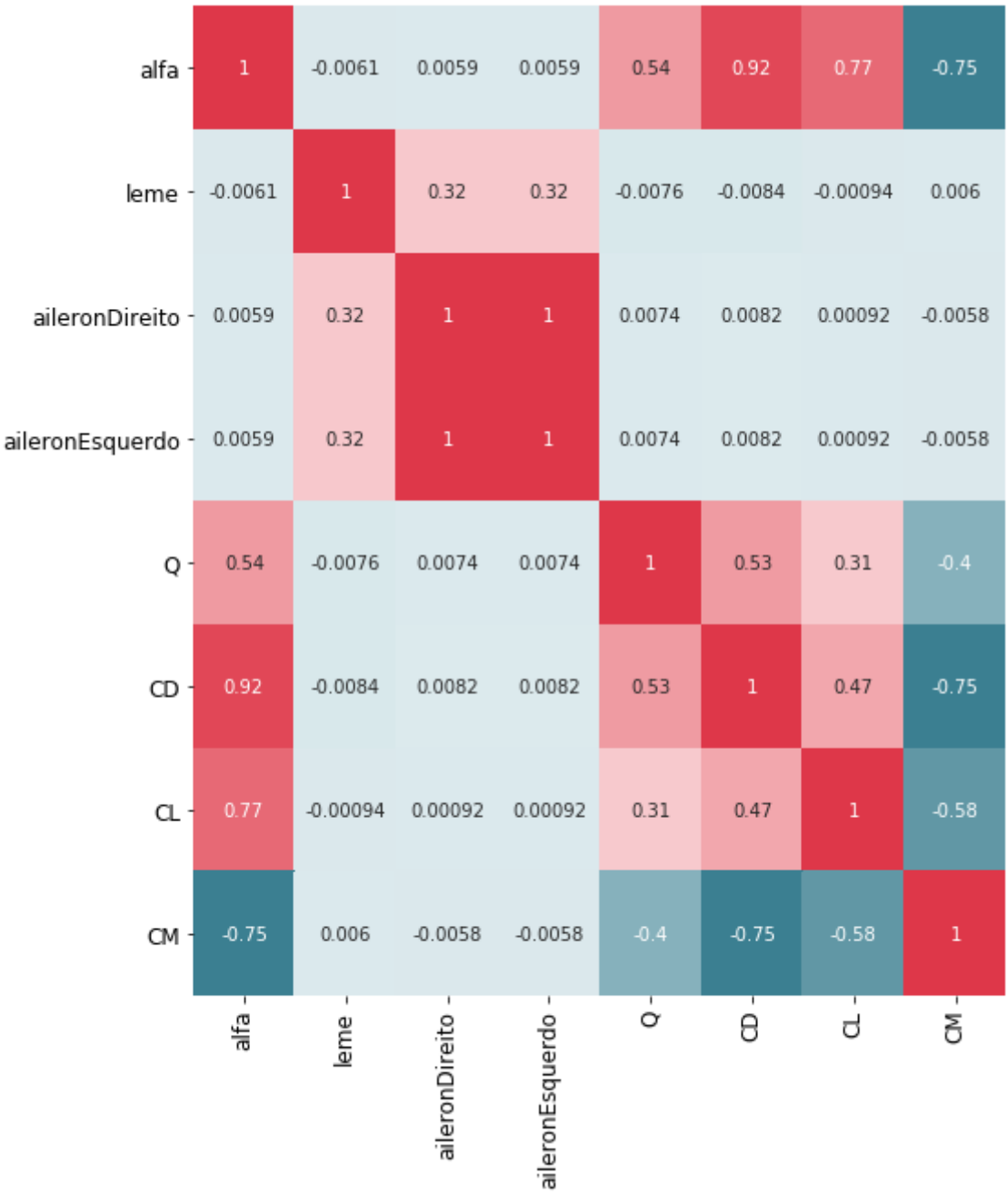


FIGURA 5.26 – Heat Map no Data Set Aterragem Alfa

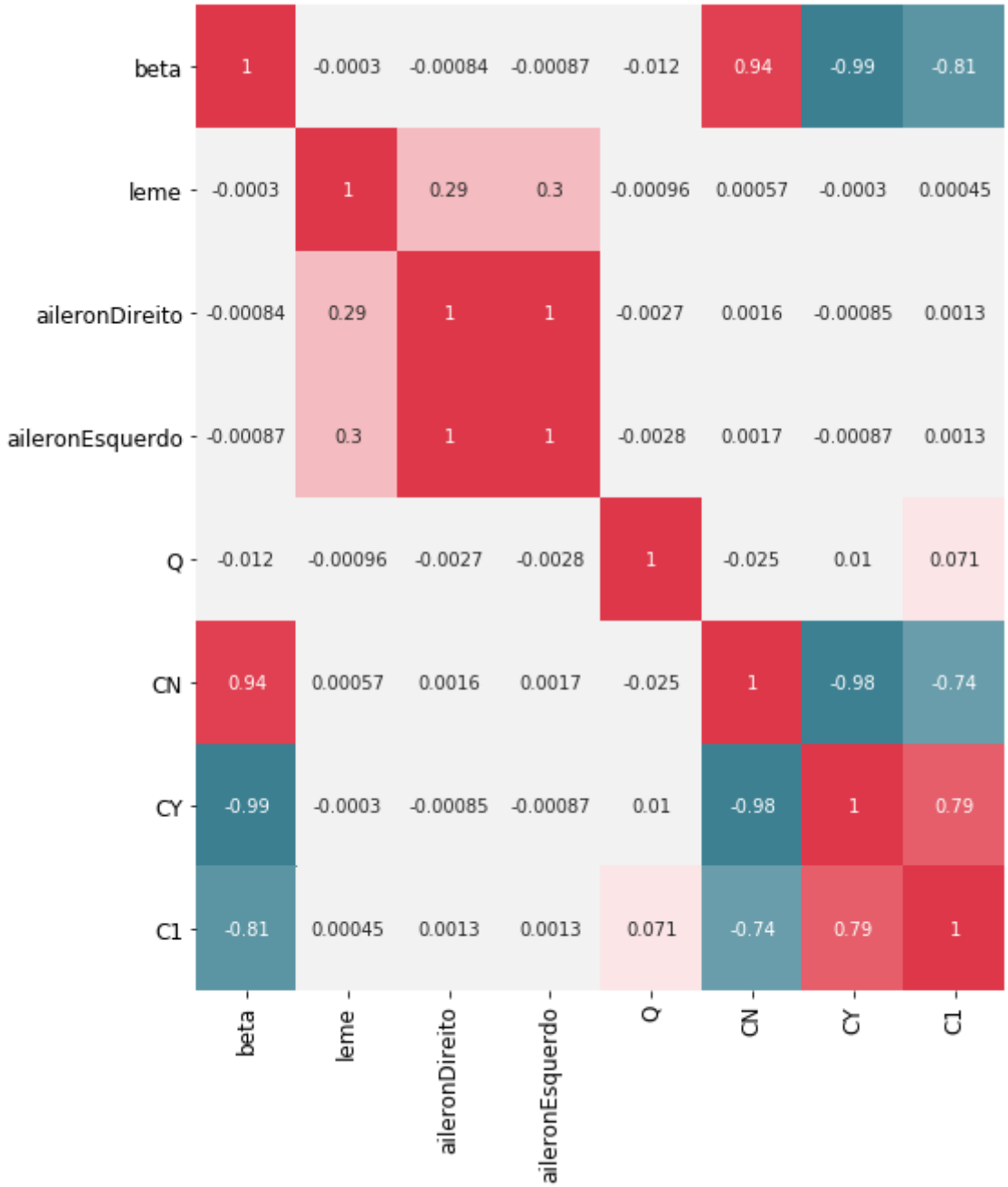


FIGURA 5.27 – Heat Map no Data Set Momentos

Em consonância com a teoria aerodinâmica, os coeficientes Aerodinâmicos dependem basicamente do ângulo de ataque α e do ângulo de guinada β (RAJKUMAR; BARDINA, 2002). É também verificada uma influência da pressão dinâmica Q nas saídas, o que está de acordo com as equações (3.1) a (3.6) .

6 Treinamento da Rede e da Random Forest

6.1 Diferenças entre os treinamentos das duas metodologias

Como já abordado no Capítulo 3, a Rede Neural requer muito mais esforço com relação à escolha dos hiper parâmetros que minimizam a função custo (*learning rate*, otimizadores, numero de neurônios por camada, funções de ativação, etc). Já com relação a *Random Forest* a única escolha é com relação ao número de árvores de decisão na floresta, o que é limitado apenas pelos recursos de memória disponíveis).

Uma diferença importante entre as metodologias é que as Redes Neurais são muito mais impactadas com diferentes configurações dos hiper parâmetros, dado que cada uma delas pode levar a um mínimo local diferente da função de custo durante otimização, o que não ocorre na *Random Forest*.

Nas seções seguintes fica evidenciado a facilidade de implementação dos treinos no algoritmo de floresta, e no caso da Rede Neural utilizou-se um procedimento mais elaborado para escolha de uma arquitetura ótima.

6.2 Treinamento das Redes Neurais

Como a amostra é restrita aos dados do ensaio, é necessário encontrar uma forma de validar o modelo apenas com os dados existentes. A abordagem utilizada é muito comum na literatura (HAN *et al.*, 2012). Ela consiste na divisão da amostra em três subconjuntos: Treino, *Cross Validation* e Teste.

- Treino: Subconjunto responsável pelo modelagem do problema, ou seja, utilizado para o cálculo dos parâmetros da função hipótese

- *Cross Validation*: Subconjunto responsável pela validação dos parâmetros encontrados pelo modelo treinado, verificando, principalmente, casos de *overfitting* e *underfitting*

- Teste: Subconjunto responsável pela validação do modelo. Por ser um conjunto de dados ainda não visto pela função hipótese, é capaz de verificar a adequação do modelo aos casos reais, indicando os erros e a medida de ajustamento do modelo

A predição dos coeficientes aerodinâmicos utilizando redes neurais foi classificado em duas categorias: Coeficientes de forças e momentos. Para forças, a entrada é o ângulo de ataque e para momentos a entrada é o angulo de guinada. Uma vez definido o conjunto de dados de treinamento, dados esparsos coletados de experimentos podem ser interpolados e estendidos para toda a gama de dados usando um treinamento em rede neural. Isso evitará repetir todos os experimentos no túnel de vento. Depois que o conjunto de dados de treinamento é selecionado, é preciso determinar o tipo de rede neural, funções de arquitetura e transferência que serão usadas para interpolar os dados esparsos.

Foram treinadas quatro redes neurais, para a modelagem de decolagemAlfa, recolhidoAlfa e aterragemAlfa e para os momentos. A tecnologia usada foi o *TensorFlow 2.0*. O máximo número de iterações para treinamento foi de 1000 *epochs*, além de ter sido programado um *Callback* para, uma vez que o treinamento estivesse bom o bastante, que se encerrasse o mesmo. Durante o treinamento a tolerância do erro foi fixada em 0.001. A taxa de aprendizagem e momentos foram selecionados apropriadamente para melhor convergência da rede. As entradas foram escaladas segundo o método *MinMax* para se assegurar que as entradas e saídas estejam em dimensões compatíveis. Quanto ao tipo de arquitetura, número de camadas, funções de ativação, número de neurônios por camada e otimizadores a serem usados , foi adotado um procedimento de tentativa e erro. Basicamente foram criadas três funções geradoras aleatória dos hiper parâmetros (número de camadas ,funções de ativação e otimizadores, sendo os parâmetros dos otimizadores mantidos fixos) e uma função que constrói o modelo a partir de tais hiper parâmetros , como a seguir :

```
1
2 def generateLayers():
3   numberLayers = randint(1,5)
4   L = []
5   for i in range(numberLayers):
6     L.append(randint(4,16))
7   return L
8
9 def generateActivations(n_layers):
```

```
10    acts = ['hard_sigmoid','relu','tanh','exponential','elu','...
       softsign']
11    A = []
12    for i in range(n_layers):
13      A.append(acts[randint(0,len(acts)-1)])
14    return A
15
16  def generateOptmizer():
17
18    op1 = keras.optimizers.SGD(learning_rate=0.01, momentum=0.0, ...
       nesterov=False)
19    op2 = keras.optimizers.RMSprop(learning_rate=0.001, rho=0.9)
20    op3 = keras.optimizers.Adagrad(learning_rate=0.01)
21    op4 = keras.optimizers.Ada△(learning_rate=1.0, rho=0.95)
22    op5 = keras.optimizers.Adam(learning_rate=0.001, beta_1=0.9, ...
       beta_2=0.999, amsgrad=False)
23    op6 = keras.optimizers.Adamax(learning_rate=0.002, beta_1=0.9, ...
       beta_2=0.999)
24    op7 = keras.optimizers.Nadam(learning_rate=0.002, beta_1=0.9, ...
       beta_2=0.999)
25    ops = [op1,op2,op3,op4,op5,op6,op7]
26    index = randint(0,len(ops)-1)
27    return ops[index],index
28
29
30  def build_model(activations,neurons,inputShape,outPutShape,opt):
31
32    l = []
33    l.append(layers.Dense(neurons[0], activation=activations[0],...
       kernel_initializer='random_normal',kernel_regularizer=keras....
       regularizers.l2(0.01), input_shape=[inputShape]))
34
35    for i in range(1,len(activations)):
36      l.append(layers.Dense(neurons[i], activation=activations[i],...
       kernel_initializer='random_normal',kernel_regularizer=keras....
       regularizers.l2(0.01)))
37
38    l.append(layers.Dense(outPutShape))
39    model = keras.Sequential(l)
40
41
42
43    model.compile(loss='mse',
44                  optimizer=opt,
45                  metrics=['mae', 'mse'])
46    return model
```

Listing 6.1 – Funções Geradoras de Hiper Parâmetros

Basicamente, a função *build model* recebe como argumento os dados aleatórios gerados pelas funções anteriores, além da dimensão de entrada e saída (*inputShape* e *outputShape*). Então, foram treinadas 100 redes neurais para cada conjunto de dados, escolhendo-se aquela que apresenta o menor erro médio quadrático no conjunto de teste. Tal procedimento só é possível dado à baixa quantidade de dados disponível, em *Datasets* maiores tal procedimento seria computacionalmente inviável. Este procedimento se assemelha a uma simulação de Monte Carlo de Otimização. O seguinte código mostra como foi feito o procedimento:

```
1
2
3 X = decolagemAlfa[['alfa']]
4 y = decolagemAlfa[['CD','CL','CM']]
5
6 train_x,test_x,train_y,test_y = train_test_split(X,y,test_size = ...
      0.25,shuffle = True)
7 train_x,test_x,train_y,test_y = np.array(train_x),np.array(test_x)...
      ,np.array(train_y),np.array(test_y)
8
9 min_mse = 10000
10 a = None
11 l = None
12 opt_index = None
13 model_decolagemAlfa = None
14 opts = ['SGD','RMSprop','Adagrad','Ada∆','Adam','Adamax','Nadam']
15 acts = ['sigmoid','relu','tanh','exponential','elu','softsign','...
      linear','softplus']
16 history = None
17 for i in range(100):
18   L = generateLayers()
19   A = generateActivations(len(L))
20   opt,ind = generateOptmizer()
21
22   model= build_model(A,L,X.shape[1],y.shape[1],opt)
23   early_stop = keras.callbacks.EarlyStopping(monitor='val_loss', ...
       patience=10)
24
25   h = model.fit(train_x, train_y, epochs=EPOCHS,validation_split =...
       0.2, verbose=0, callbacks=[early_stop])
26
27   loss, mae, mse = model.evaluate(test_x, test_y, verbose=2)
```

```
28    if mse < min_mse:
29        min_mse = mse
30        a = A
31        l = len(L)
32        opt_index = ind
33        model_decolagemAlfa = model
34        history = h
35
36
37 print (model_decolagemAlfa.summary())
38 print(l)
39 print (a)
40 print (opts[opt_index])
41 plot_history(history)
```

Listing 6.2 – Treinamento de 100 redes neurais e armazenando o modelo com menor erro quadrático

A título de observação, não necessariamente o modelo escolhido é o que melhor modela os dados, porém os modelos mostraram-se satisfatórios. A seguir é mostrado os parâmetros escolhidos para cada conjunto de dados. Destaca-se a rápida convergência de aprendizagem para tais parâmetros.

1. decolagemAlfa
   (a) Número de Camadas : 2
   (b) Número de Neurons/Camada: 15 e 6
   (c) Funções de Ativação : Exponential e Softsign
   (d) Otimizador : Adagrad
2. recolhidoAlfa
   (a) Número de Camadas : 3
   (b) Número de Neurons/Camada: 6,9 e 4
   (c) Funções de Ativação : Exponential , Tanh e Softsign
   (d) Otimizador : Adam
3. aterragemAlfa
   (a) Número de Camadas : 1
   (b) Número de Neurons/Camada: 9
   (c) Funções de Ativação : Relu

(d) Otimizador : SGD

4. Momentos

(a) Número de Camadas : 1

(b) Número de Neurons/Camada: 6

(c) Funções de Ativação : Tanh

(d) Otimizador : Adagrad

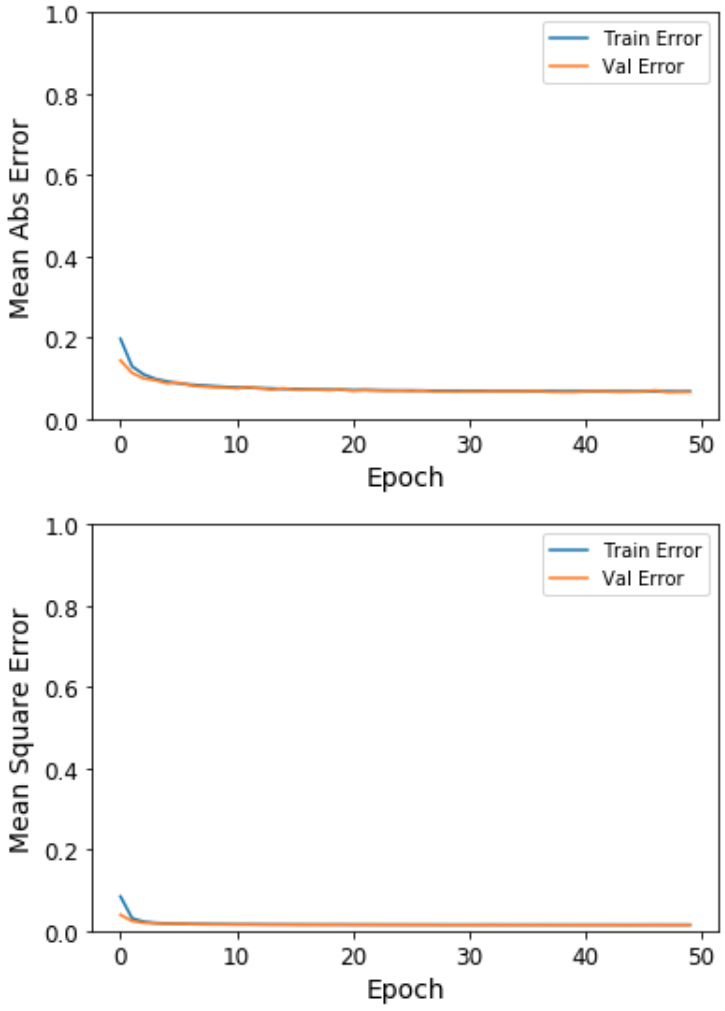


FIGURA 6.1 – Curva de aprendizado para os coeficientes de força da aeronave em decolagem.

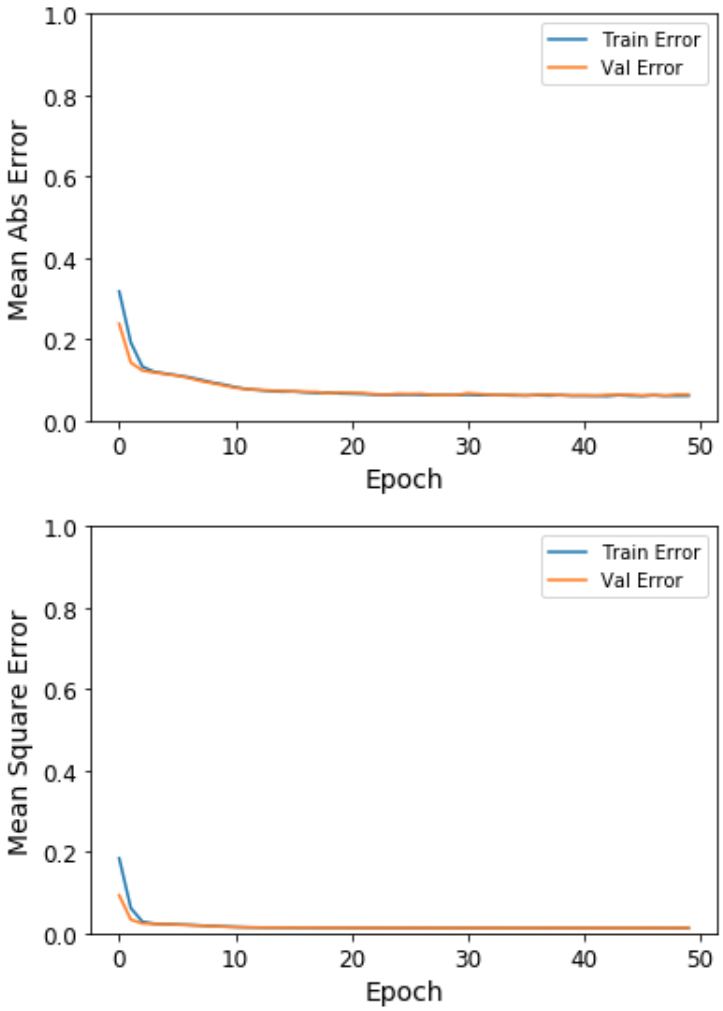


FIGURA 6.2 – Curva de aprendizado para os coeficientes de força da aeronave com trem de pouso recolhido

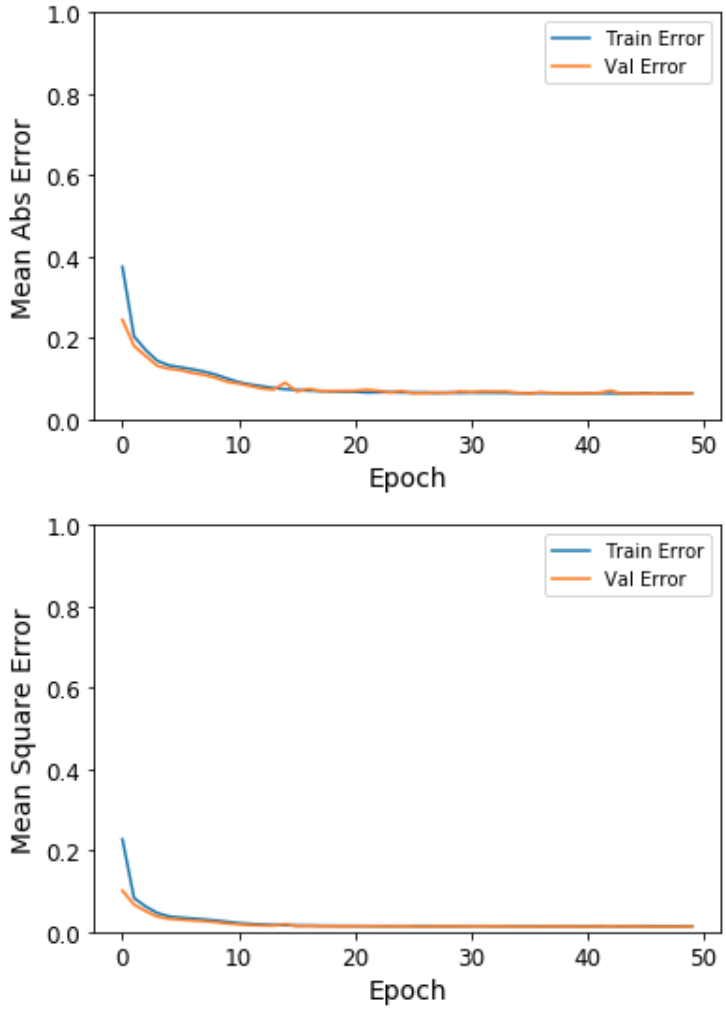


FIGURA 6.3 – Curva de aprendizado para os coeficientes de força da aeronave em aterrissagem.

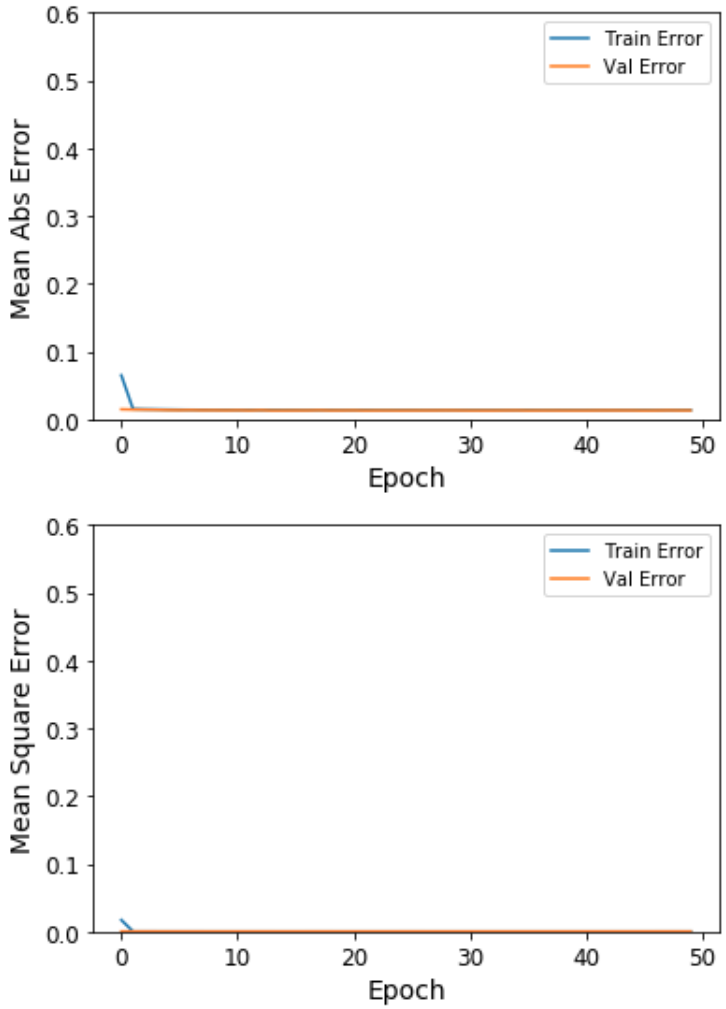


FIGURA 6.4 – Curva de aprendizado para os coeficientes de momento da aeronave

6.3 Random Forest

A modelagem usando *Random Forest* apresentou-se bastante satisfatória e representativa dos modelos aerodinâmicos, além de sua implementação ser relativamente simples, onde há apenas 1 Hiper Parâmetro a considerar, o número de árvores de decisão usado para modelagem do problema, ao contrário das redes neurais. No presente trabalho, foram usada 100 árvores de decisão. O número de árvores não acarreta problemas de *overfitting*, o custo vêm apenas em termos de recursos de memória. Como não se trata de um algoritmo que necessita de uma função distância entre vetores, para cálculo de erros por exemplo, todas as *features* foram consideradas . Mesmo que alguma *feature* não seja importante para cálculo de determinado coeficiente, a metodologia *bagging* descrita no Capítulo 3 assegura que isso não afetará a predição, diferentemente da Rede Neural. Mesmo assim, ficou evidente que de fato os parâmetros α e β são os mais essenciais para predição. A seguir é apresentado a modelagem para um dos conjuntos de dados:

```
1
2 from sklearn.ensemble import RandomForestRegressor
3 from sklearn.metrics import mean_squared_error
4 from sklearn.preprocessing import MinMaxScaler
5
6
7 X = aterragemAlfa[['alfa','profundor','leme','aileronDireito','...
    aileronEsquerdo','Q']]
8 y = aterragemAlfa[['CD','CL','CM']]
9
10
11 train_x,test_x,train_y,test_y = train_test_split(X,y,test_size = ...
    0.25,random_state = 11,shuffle = True)
12 train_x,test_x,train_y,test_y = np.array(train_x),np.array(test_x)...
    ,np.array(train_y),np.array(test_y)
13
14
15 forest_reg = RandomForestRegressor(n_estimators= 1000)
16 forest_reg.fit(train_x,train_y)
17 predictions = forest_reg.predict(test_x)
18 forest_mse = mean_squared_error(test_y,predictions)
19 forest_rmse = np.sqrt(forest_mse)
```

Listing 6.3 – Treinando Random Forests

E com o método *feature importance* incluído na Floresta, temos um resultado parecido com as Correlações vistas no Capítulo 5. Para a aeronave com trem de pouso recolhido, temos as seguintes importâncias das entradas para a predição das saídas C_D, C_L e C_M :

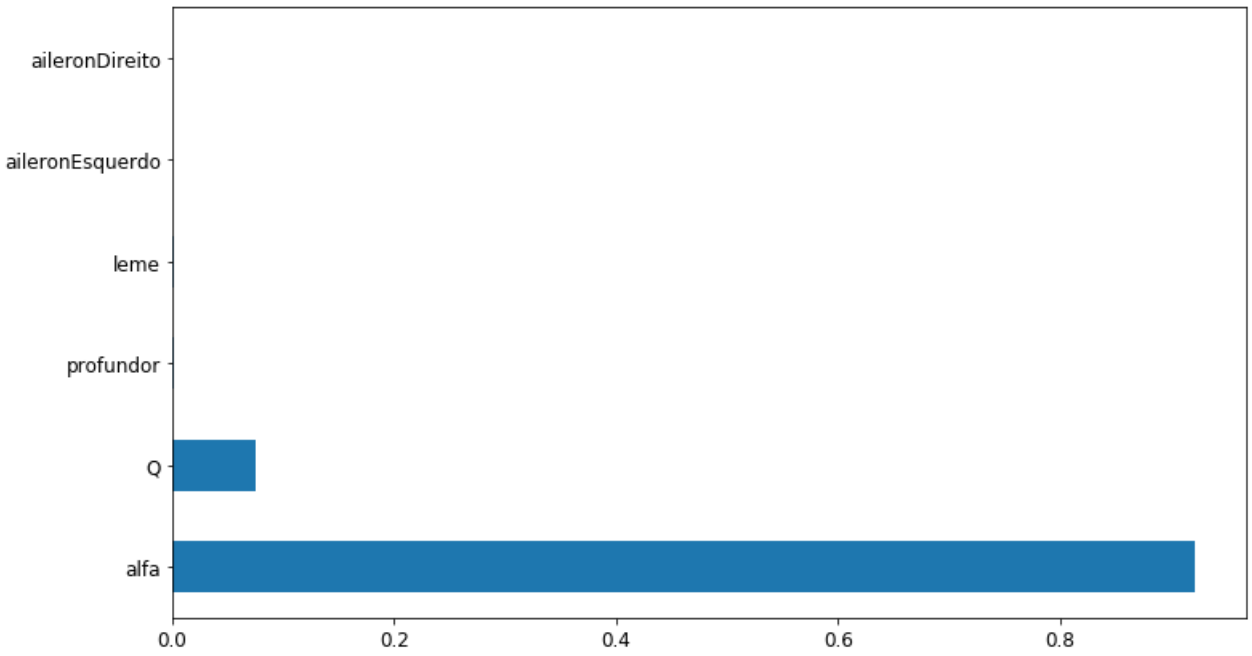


FIGURA 6.5 – Feature Importance da Floresta Aleatória.

7 Resultados e Discussões

7.1 Modelo da Rede Neural

Para a Rede, C_M é o mais difícil de prever em comparação com o coeficiente de sustentação e arrasto, dos quais algumas amostras de teste na região próxima a zero são previstas muito acima da margem de erro de 5%. Tal resultado se aproxima muito da literatura de estudos semelhantes (ZHANG;XINGUO,2018). As arquiteturas usadas foram as descritas no Capítulo 6. Os erros também mostram-se satisfatórios, por tenderem a zero. A seguir são mostrados as distribuições dos erros de predição e a comparação entre valores reais e previstos.

Decolagem Alfa

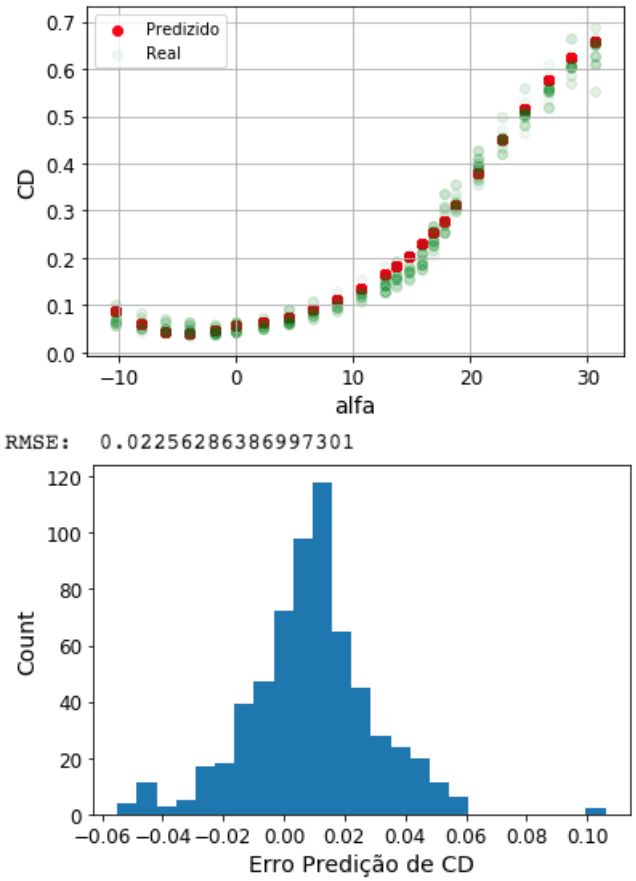


FIGURA 7.1 – Predição da Rede Neural de C_d em decolagem

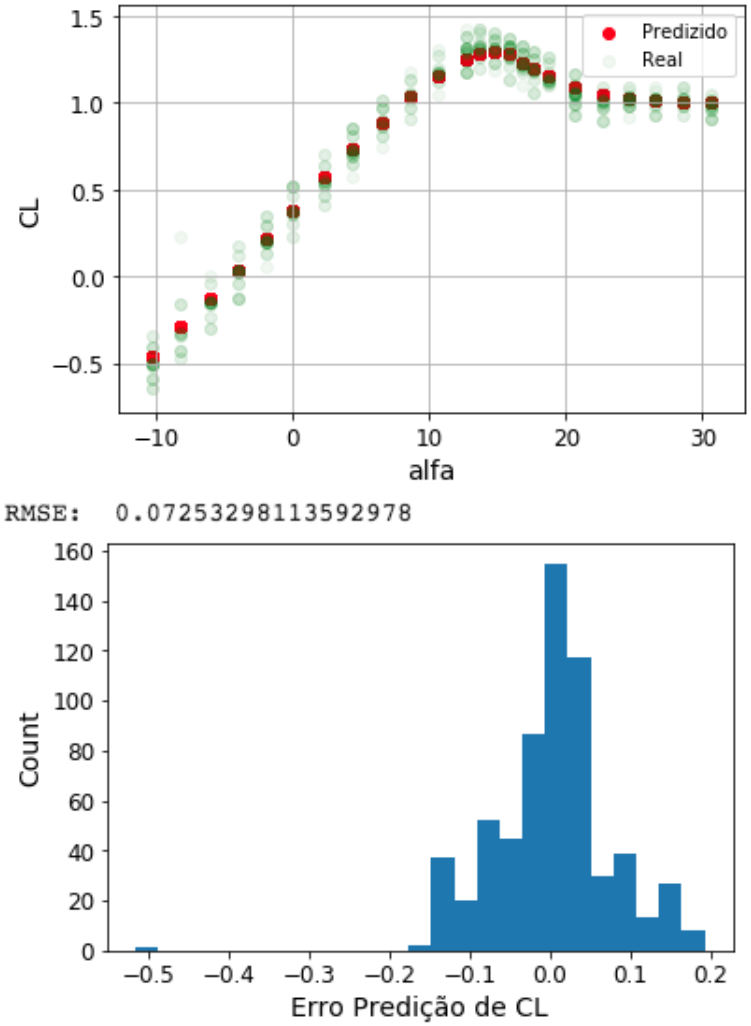


FIGURA 7.2 – Predição da Rede Neural de C_l em decolagem

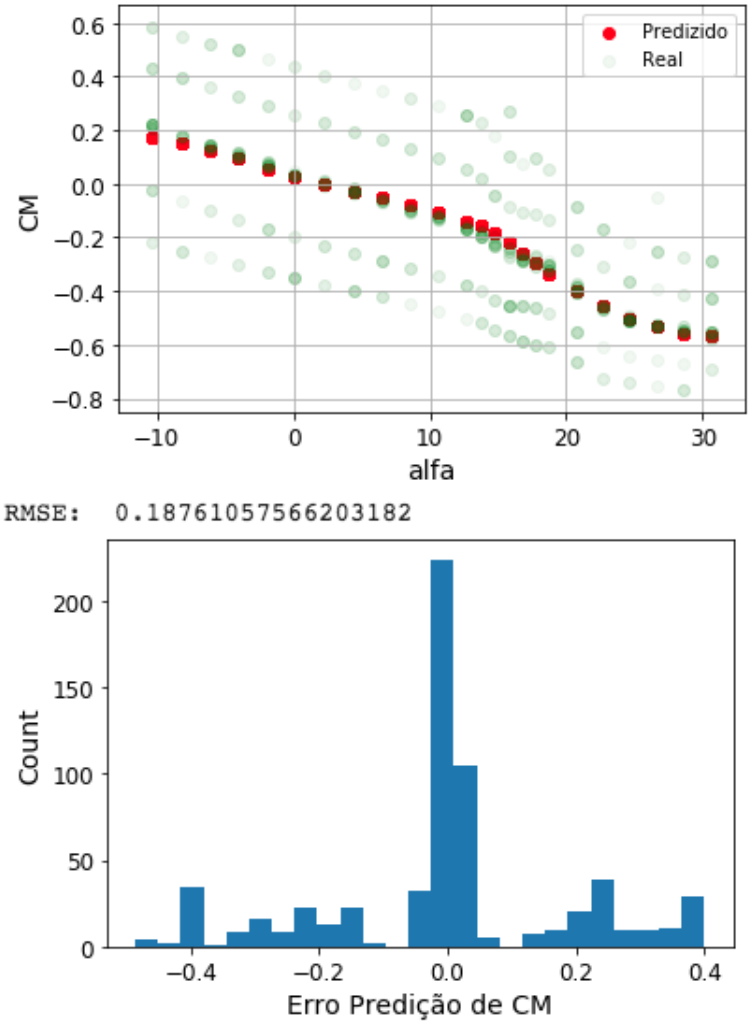


FIGURA 7.3 – Predição da Rede Neural de C_m em decolagem

Recolhido Alfa

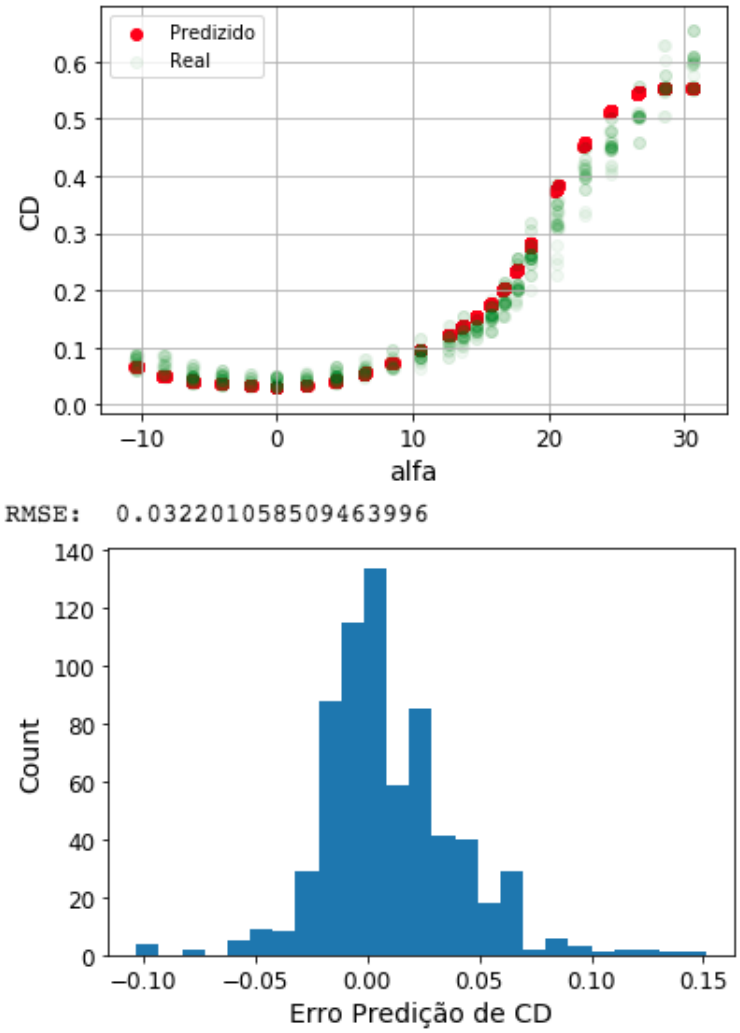


FIGURA 7.4 – Predição da Rede Neural de C_d com trem de pouso recolhido

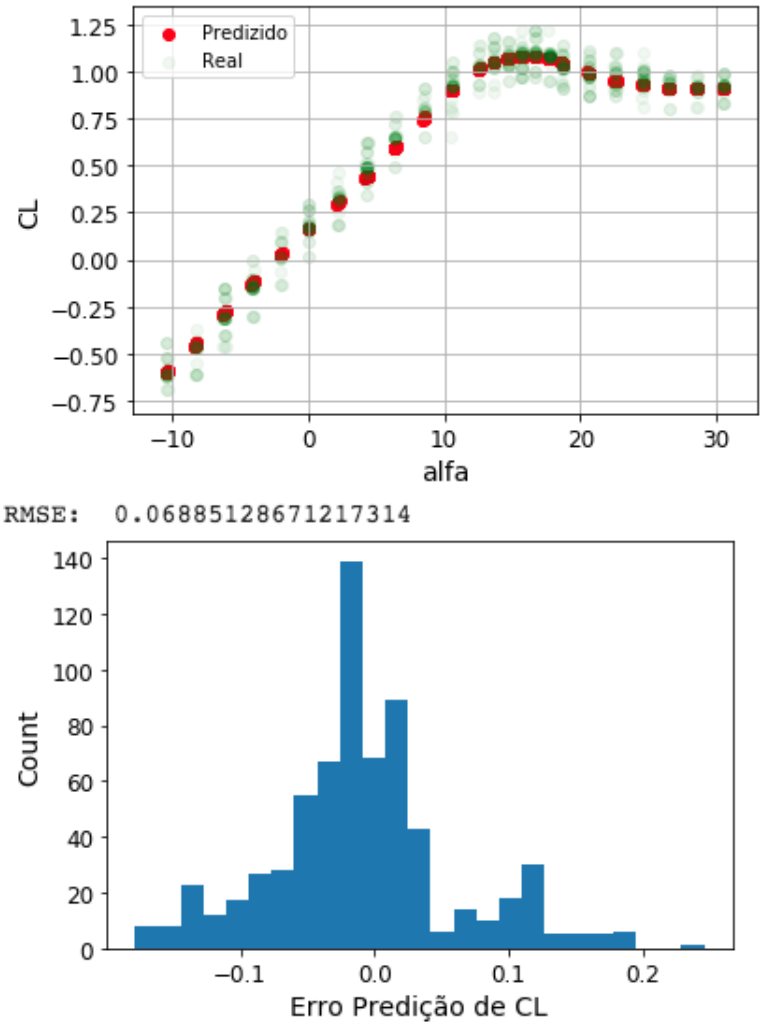


FIGURA 7.5 – Predição da Rede Neural de C_l com trem de pouso recolhido

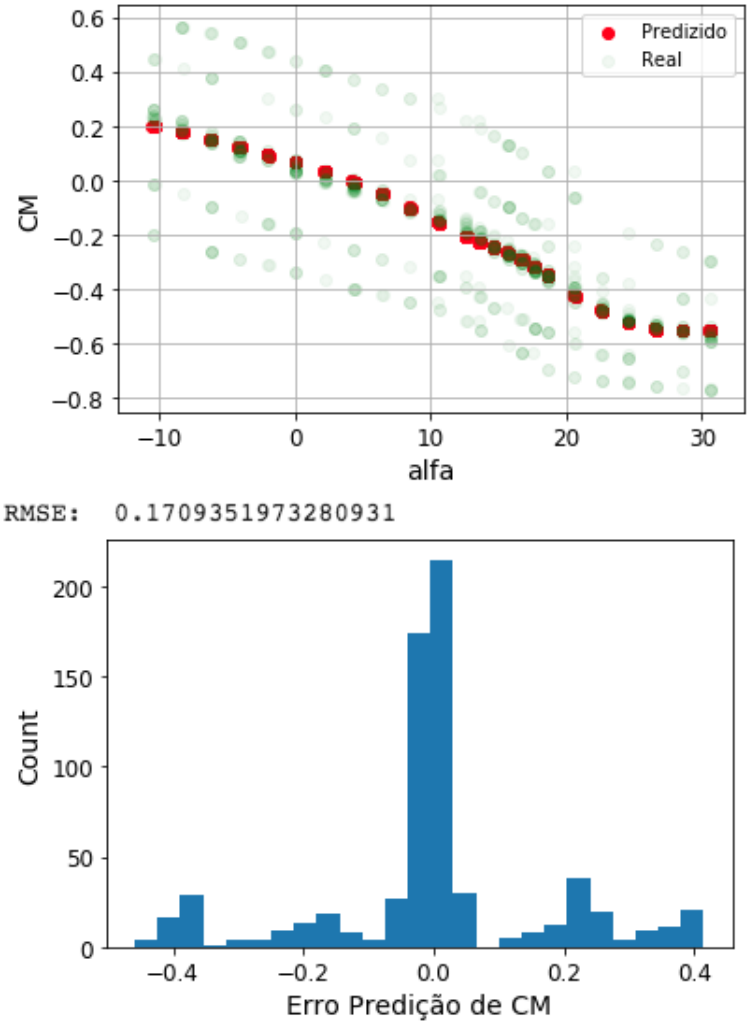


FIGURA 7.6 – Predição da Rede Neural de C_m com trem de pouso recolhido

Aterrissagem Alfa

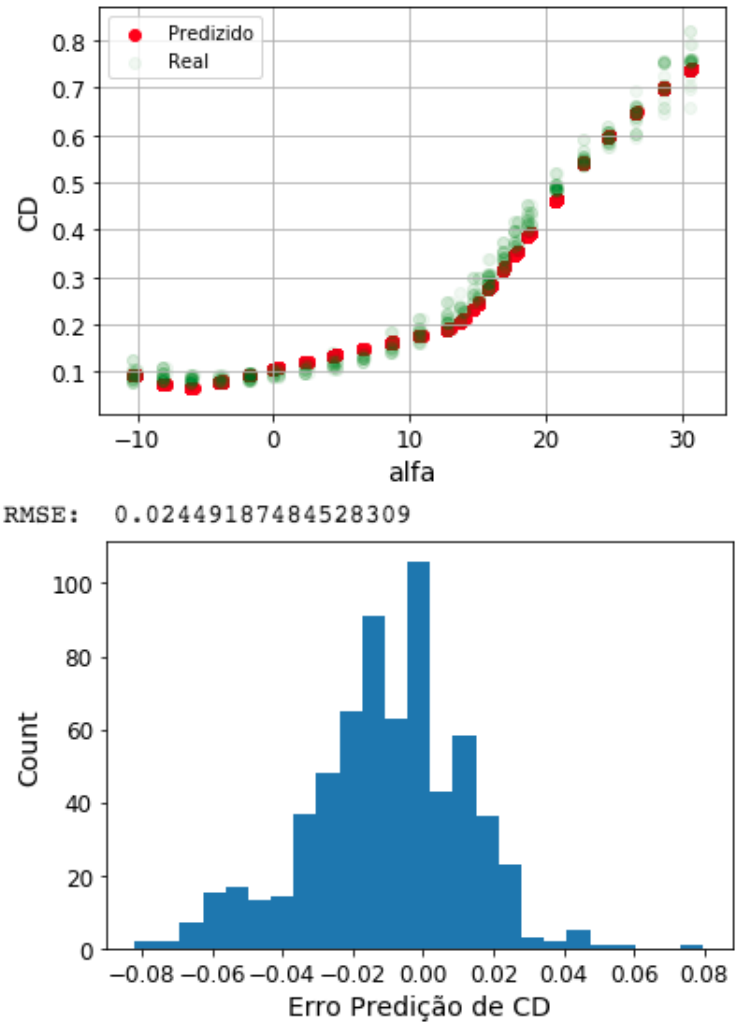


FIGURA 7.7 – Predição da Rede Neural de C_d em aterrissagem

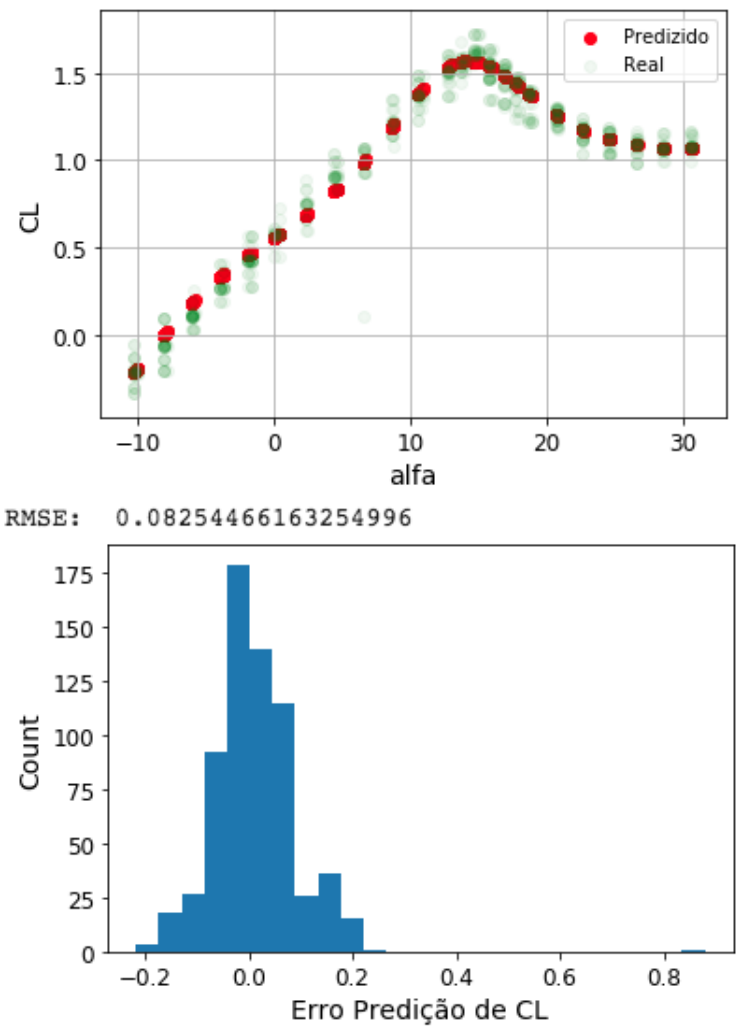


FIGURA 7.8 – Predição da Rede Neural de C_l em aterrissagem

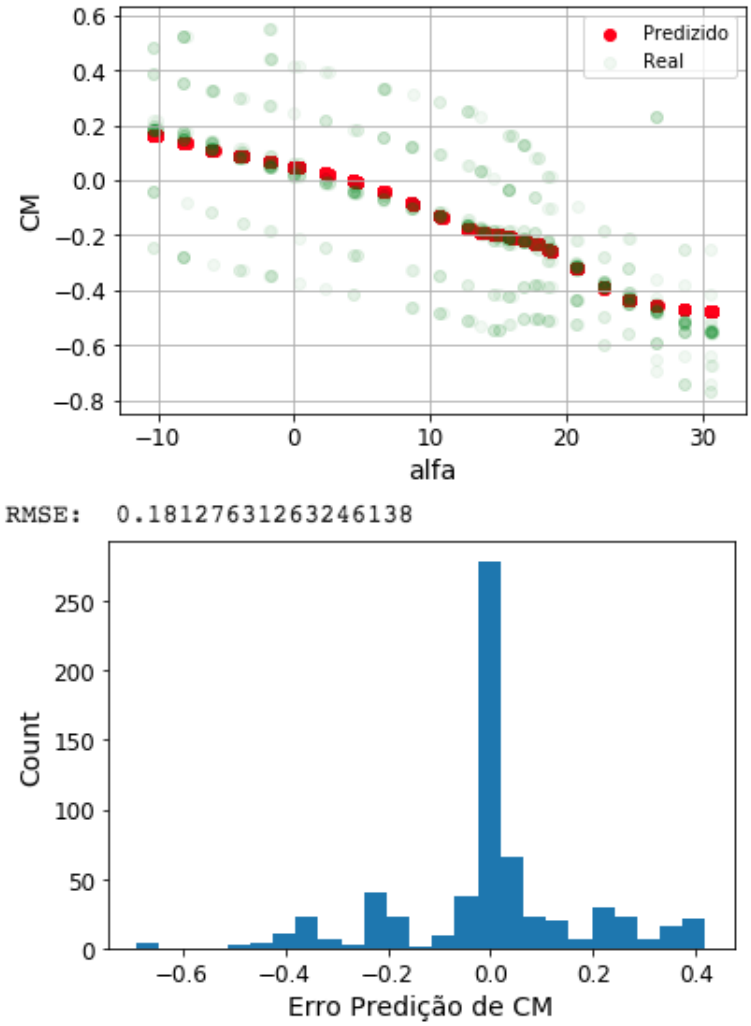


FIGURA 7.9 – Predição da Rede Neural de C_m em aterrissagem

Momentos

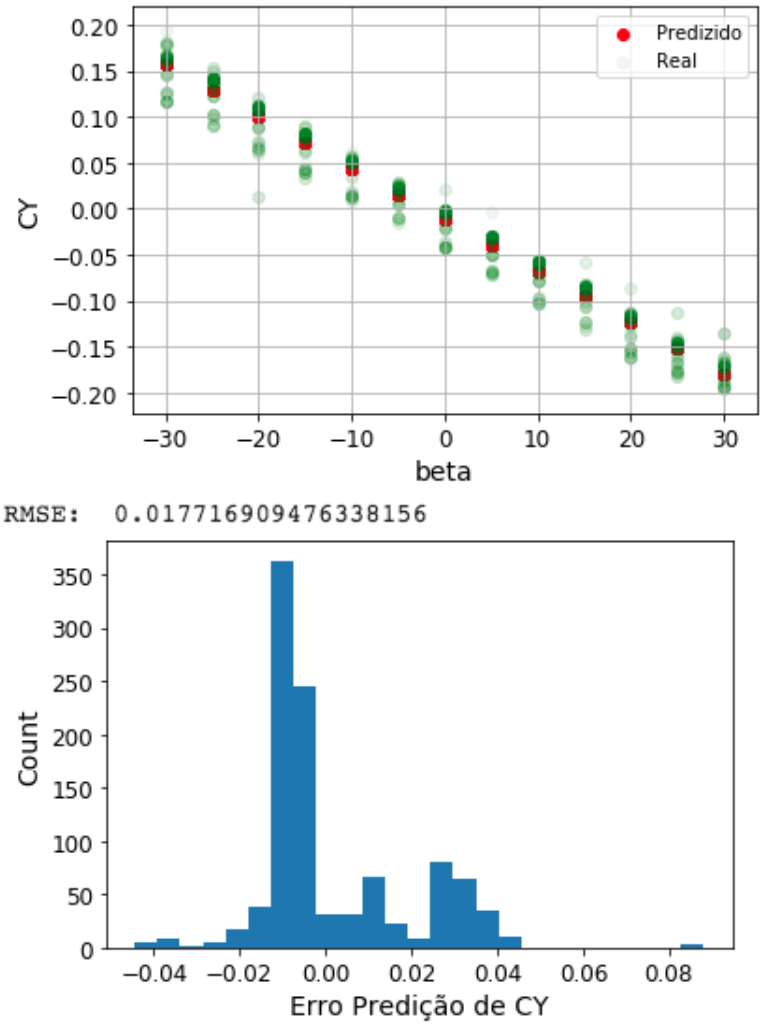


FIGURA 7.10 – Predição da Rede Neural de C_y

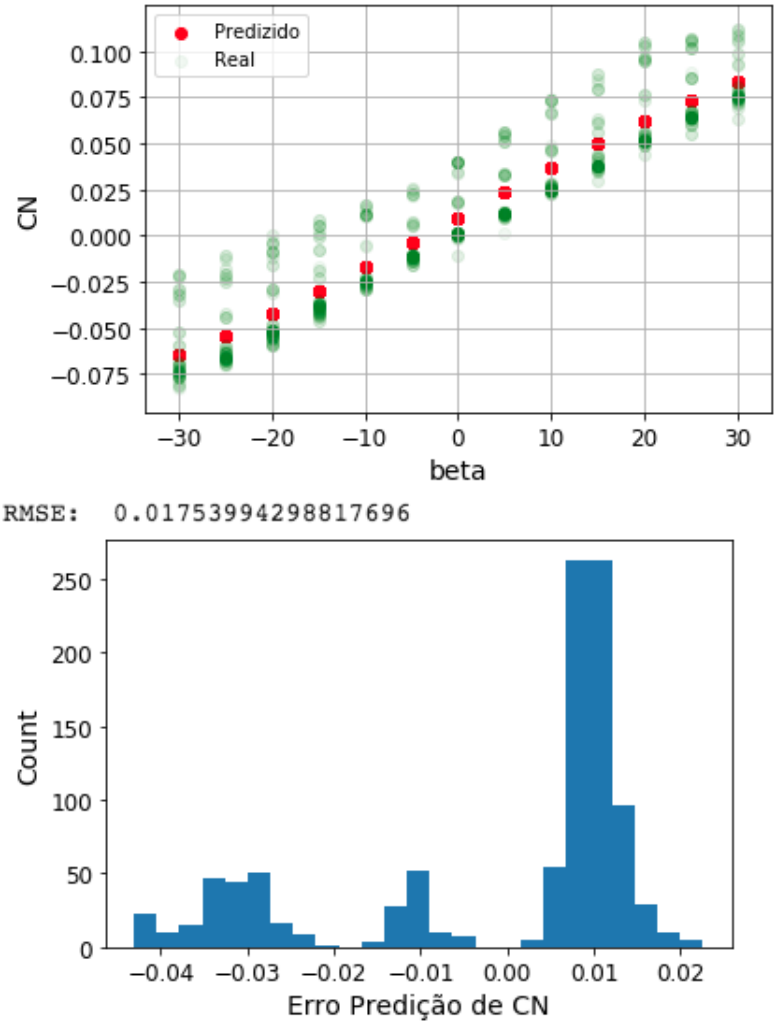


FIGURA 7.11 – Predição da Rede Neural de C_n

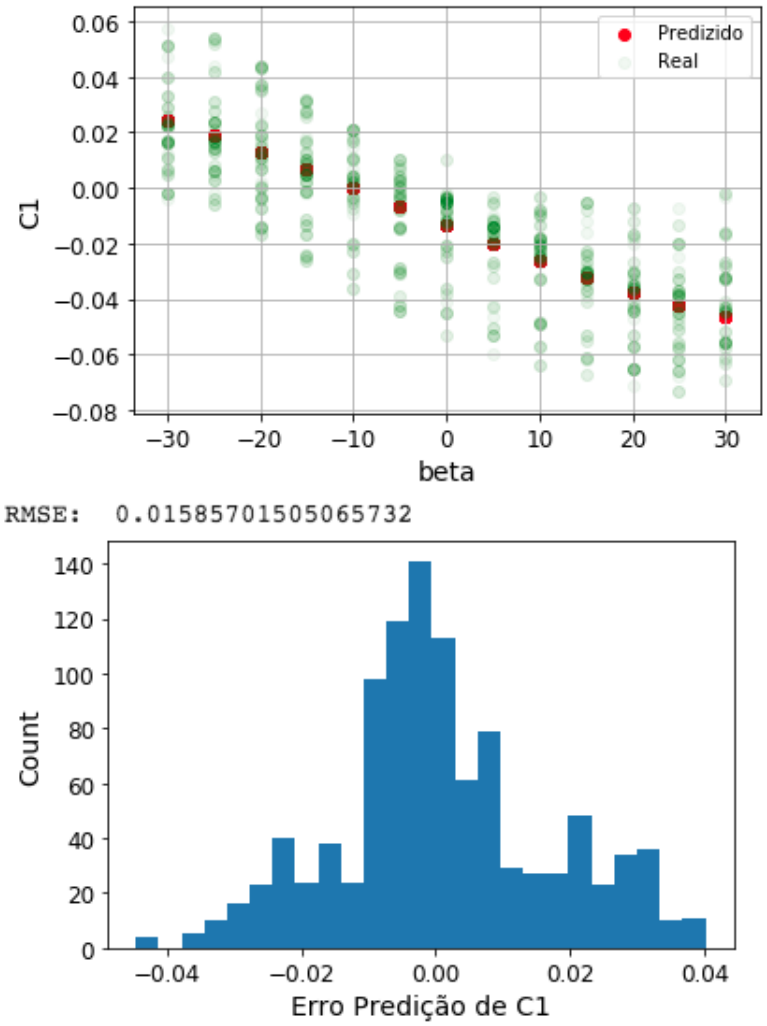


FIGURA 7.12 – Predição da Rede Neural de C_1

7.2 Modelo de Random Forest

A metodologia de Floresta mostrou-se bastante representativa dos coeficientes, em especial com relação a C_M e aos coeficientes de momento, C_N, C_Y, C_1, os quais a rede apresentou maior dificuldade de predição. A seguir são mostrados as distribuições dos erros de predição e a comparação entre valores reais e previstos. Graficamente é possível perceber a melhor performance da *Random Forest* em relação a Rede.

Decolagem Alfa

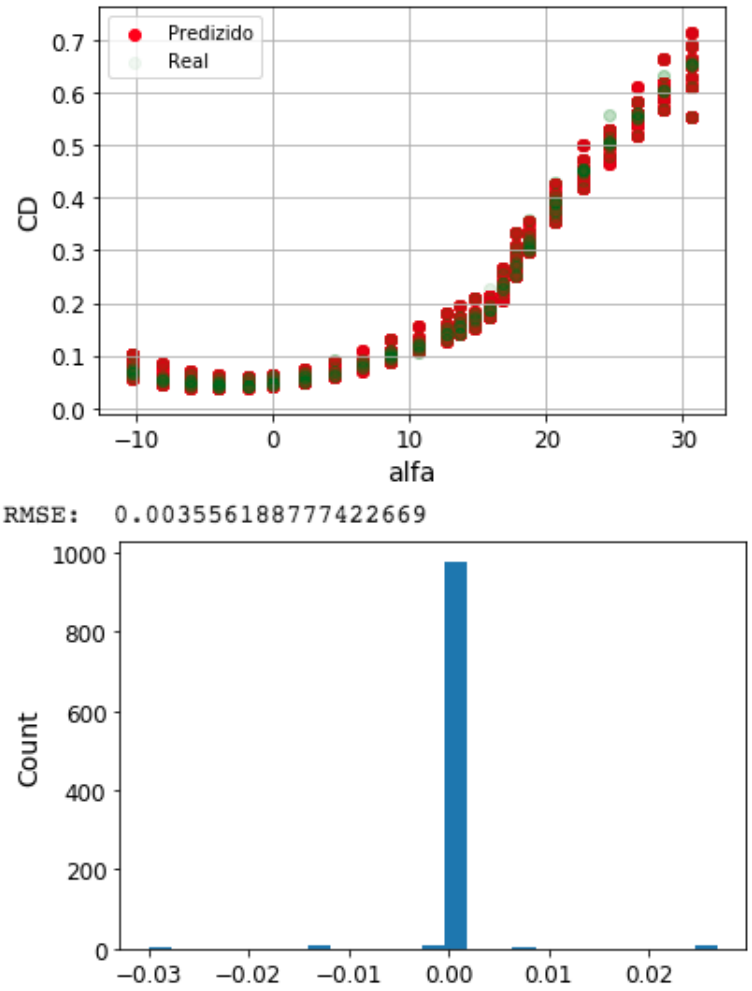


FIGURA 7.13 – Predição da Random Forest de C_d em decolagem

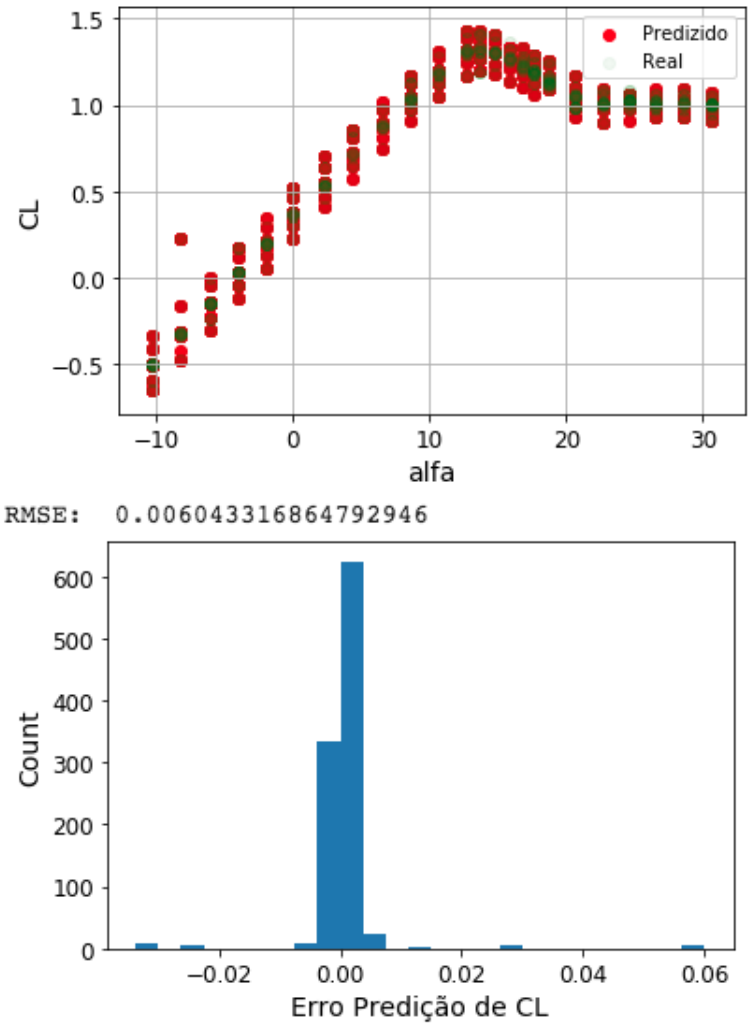


FIGURA 7.14 – Predição da Random Forest de C_l em decolagem

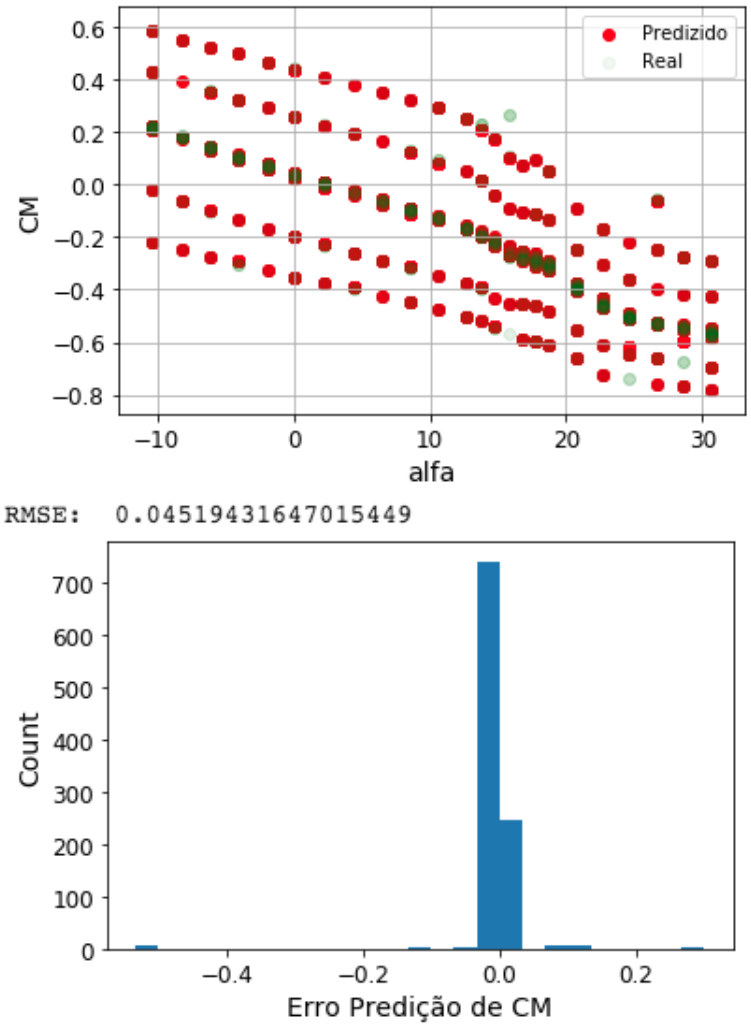


FIGURA 7.15 – Predição da Random Forest de C_m em decolagem

Recolhido Alfa

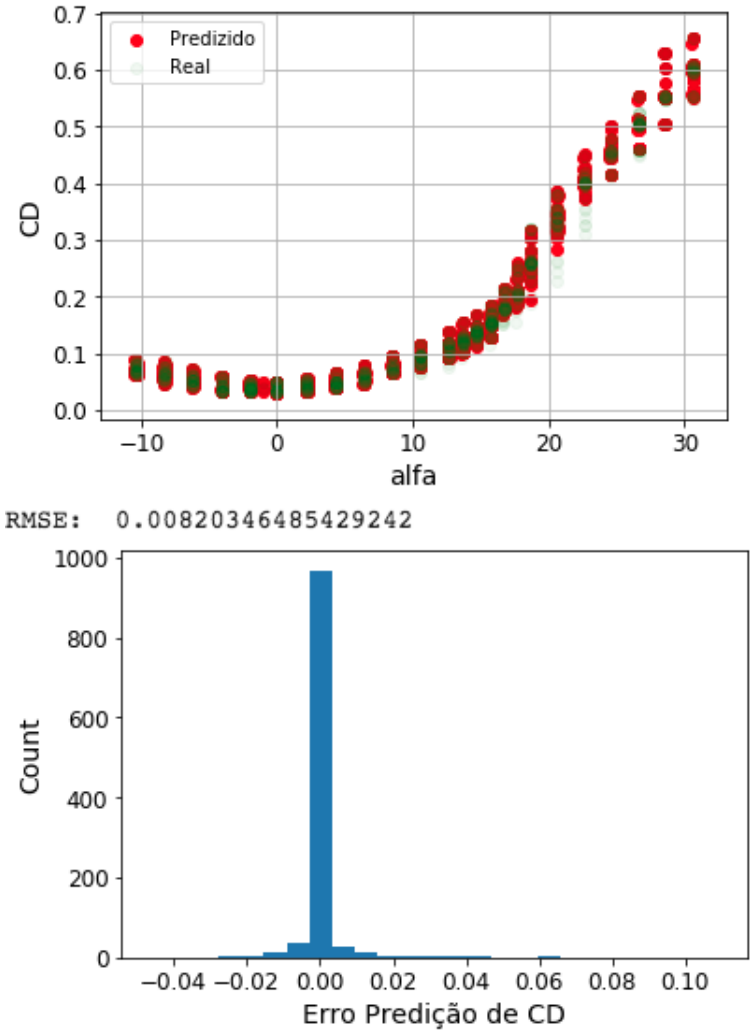


FIGURA 7.16 – Predição da Random Forest de C_d com trem de pouso recolhido

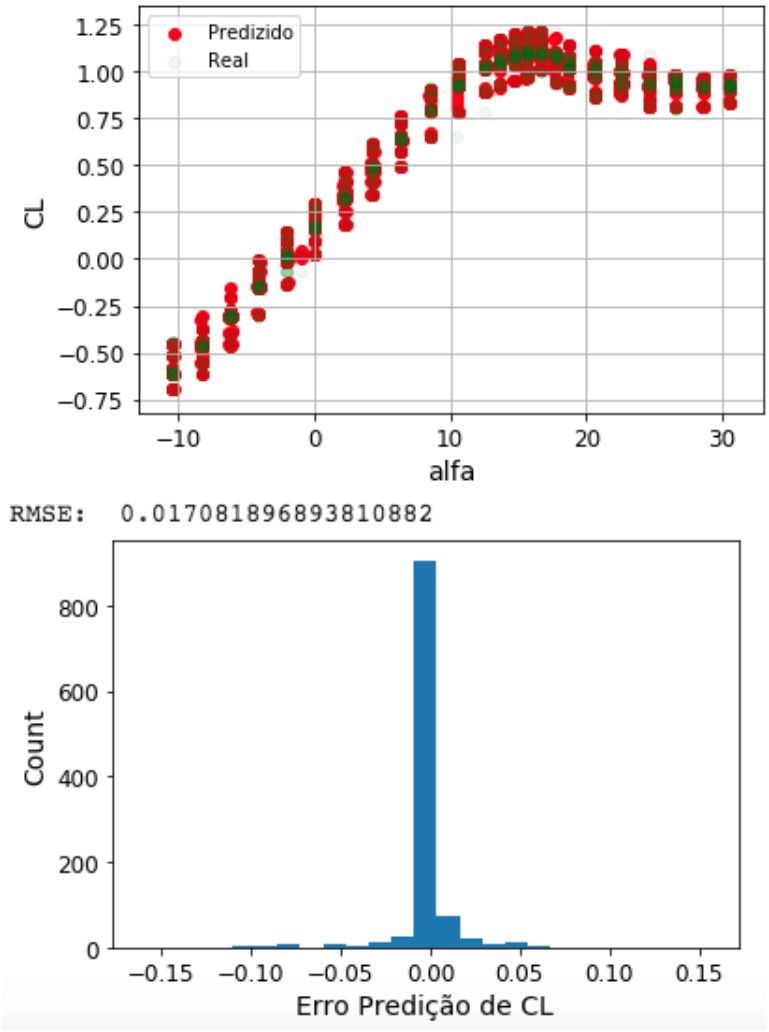


FIGURA 7.17 – Predição da Random Forest de C_l com trem de pouso recolhido

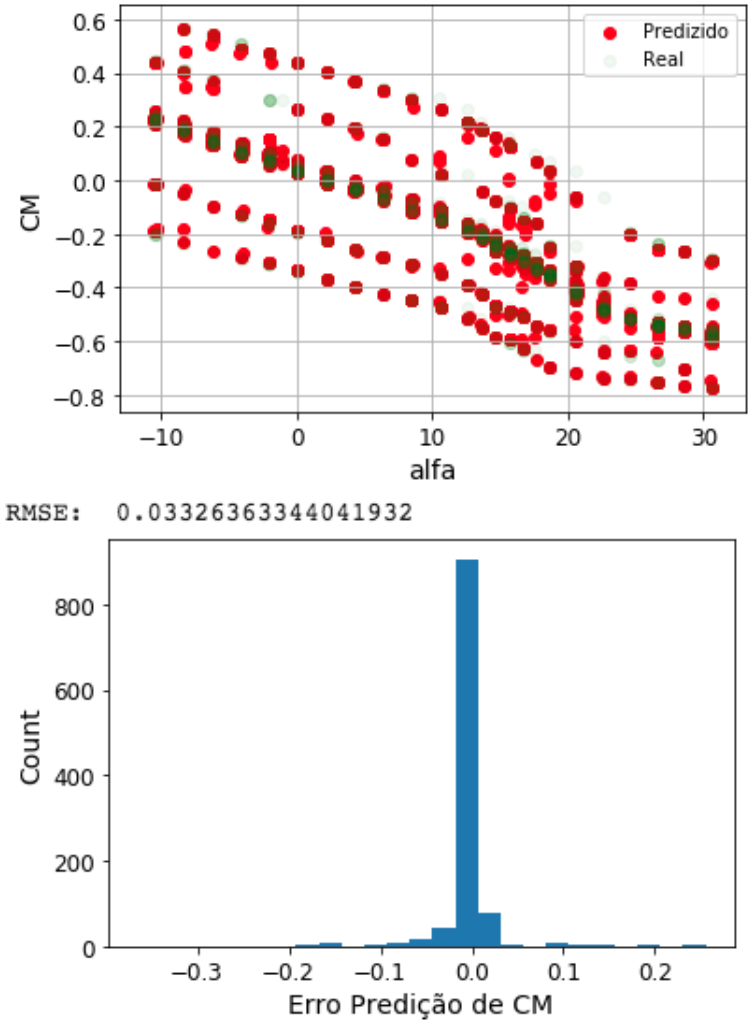


FIGURA 7.18 – Predição da Random Forest de C_m com trem de pouso recolhido

Aterrissagem Alfa

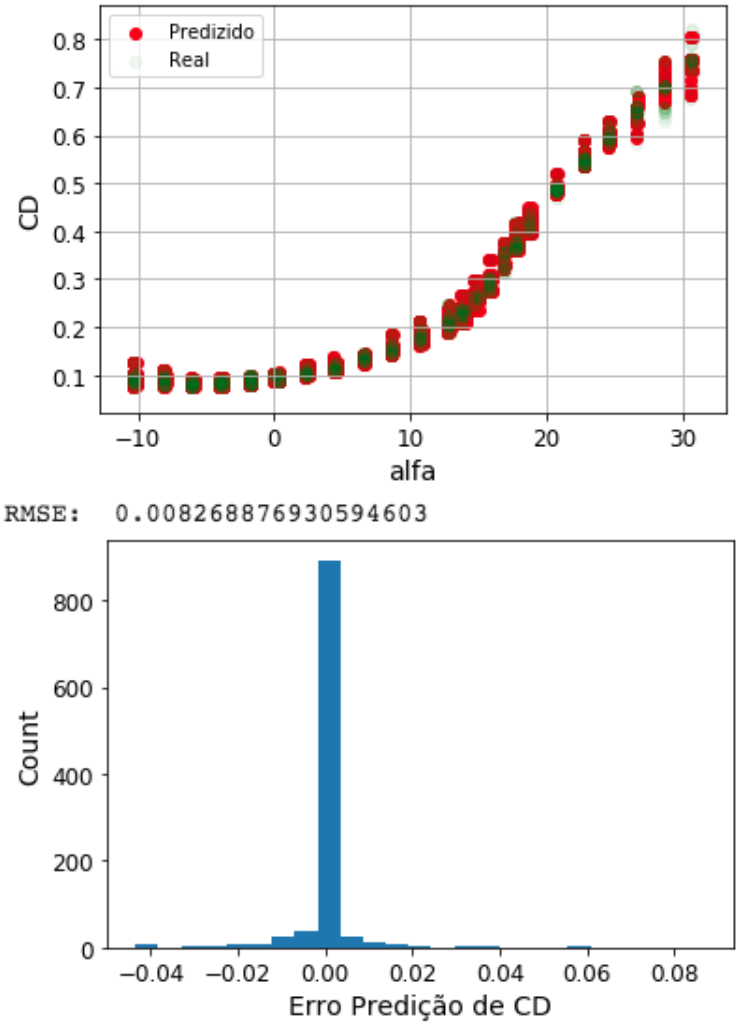


FIGURA 7.19 – Predição da Random Forest de C_d em aterrissagem

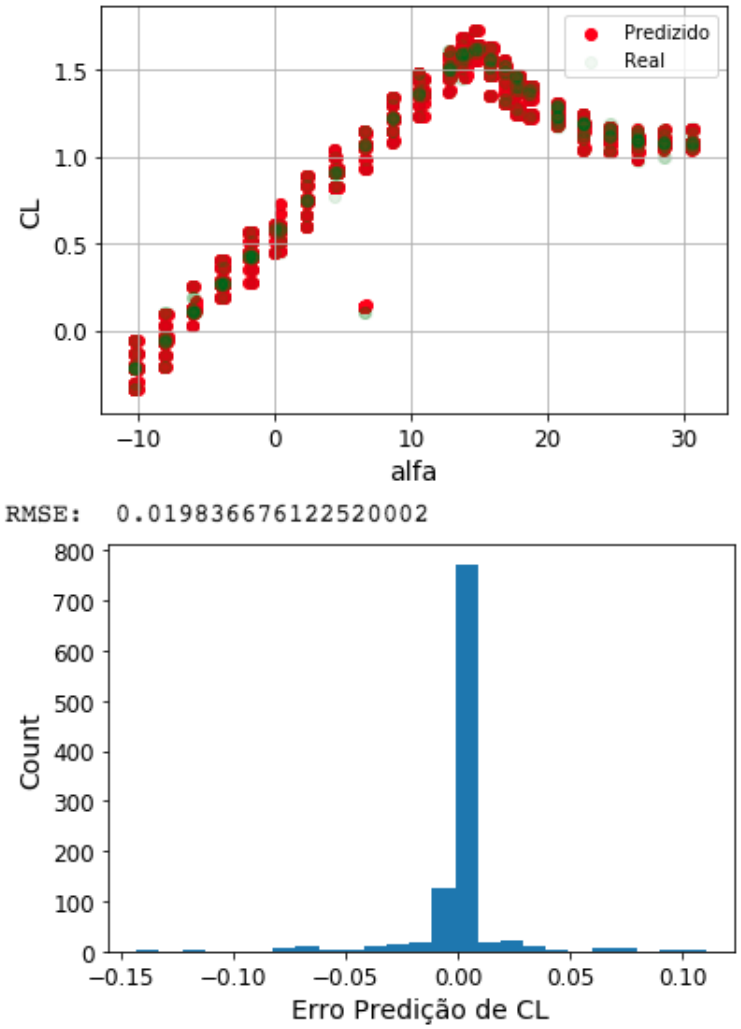


FIGURA 7.20 – Predição da Random Forest de C_l em aterrissagem

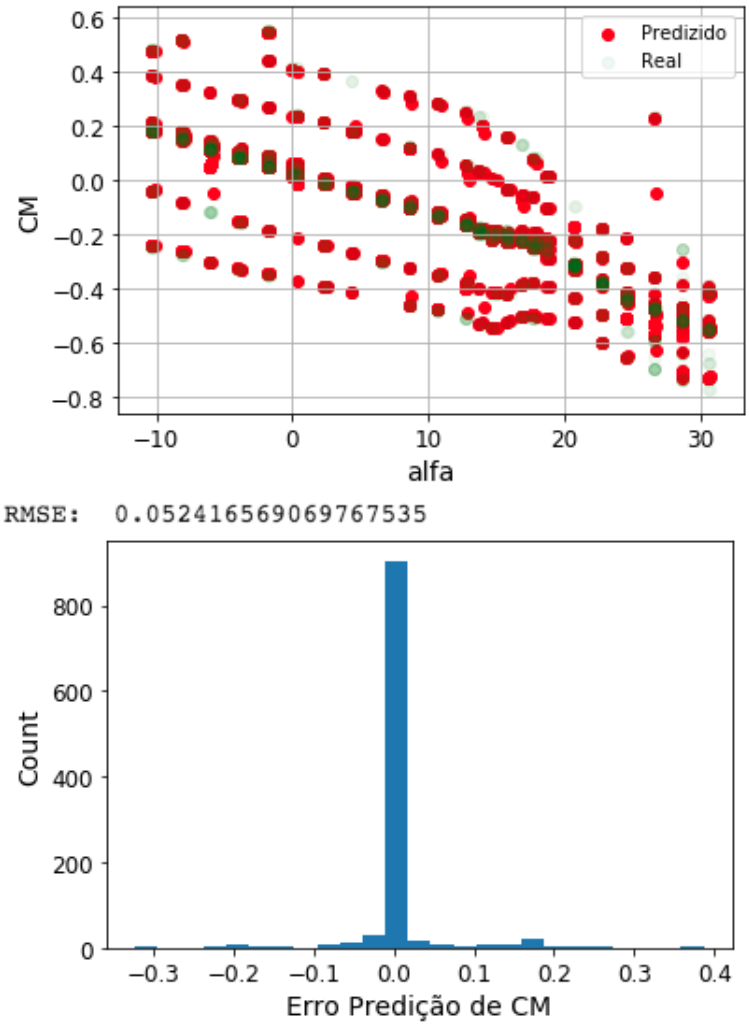


FIGURA 7.21 – Predição da Random Forest de C_m em aterrissagem

Momentos

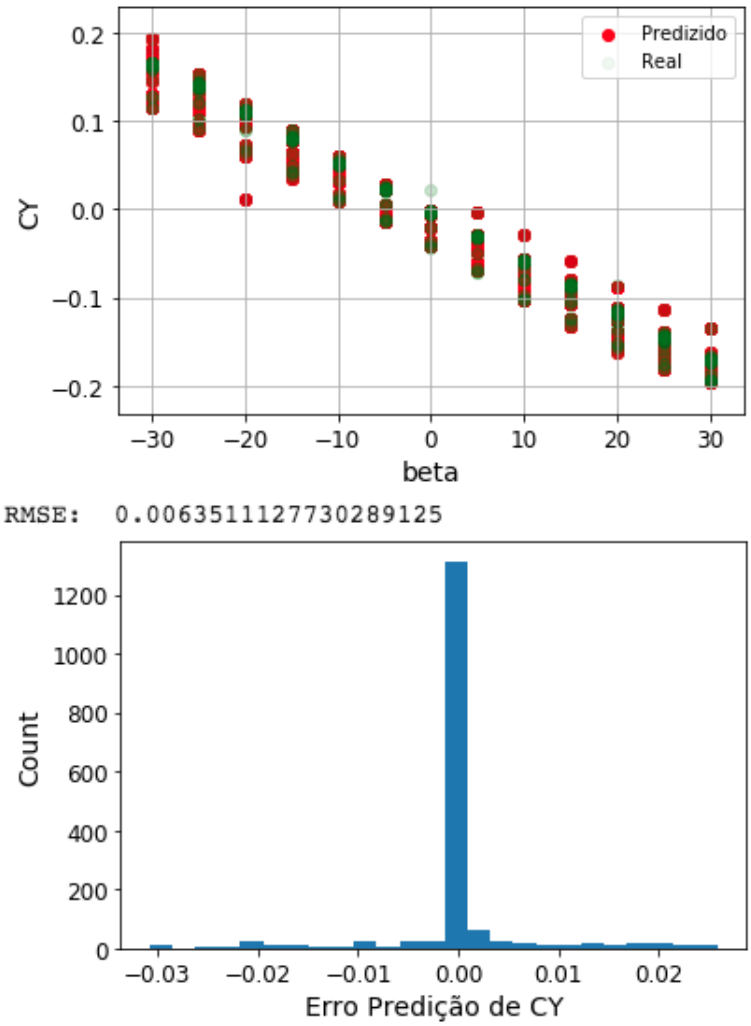


FIGURA 7.22 – Predição da Random Forest de C_y

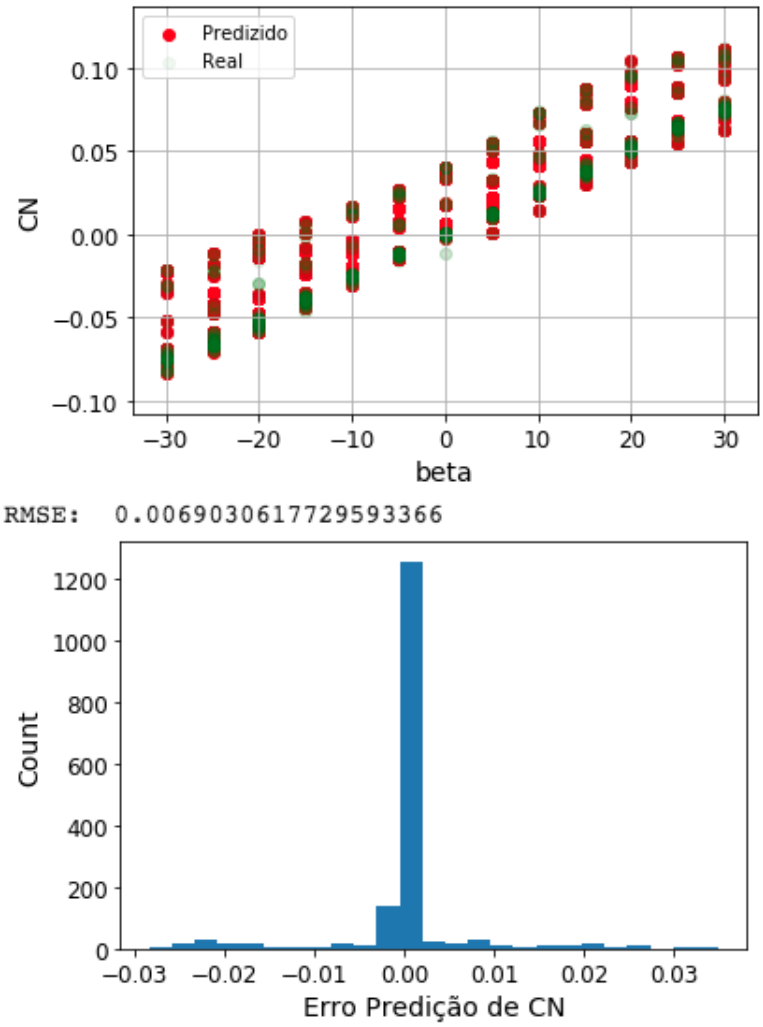


FIGURA 7.23 – Predição da Random Forest de C_n

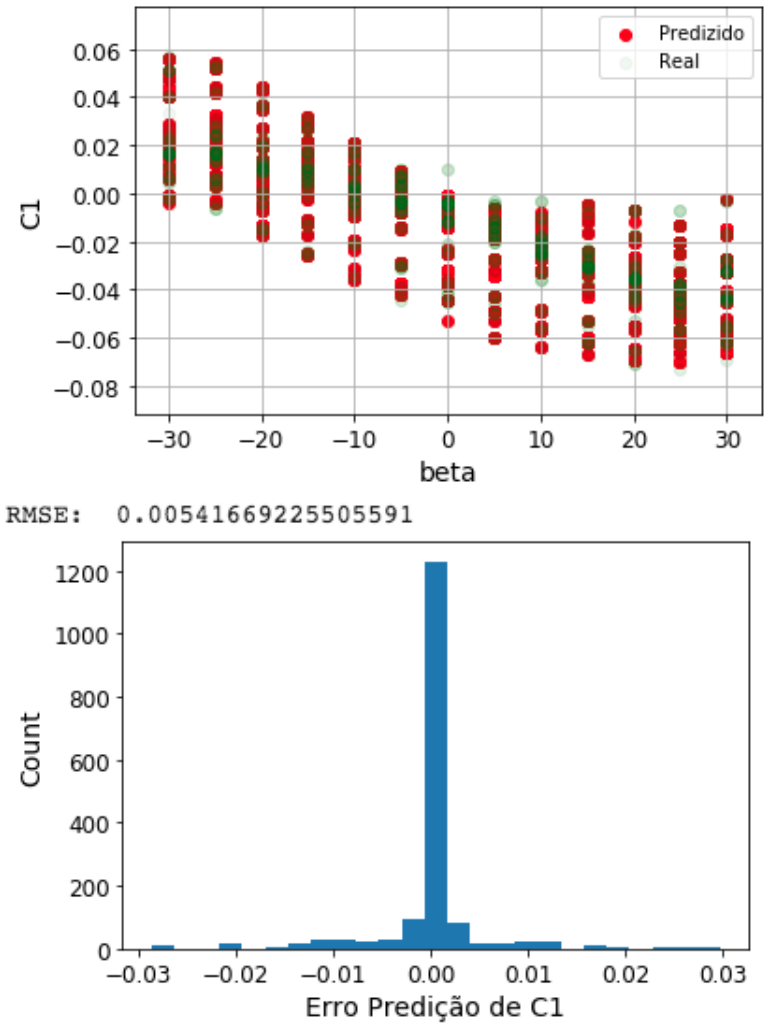


FIGURA 7.24 – Predição da Random Forest de C_1

A primeira observação é quanto à distribuição dos erros. A predição na floresta possuem média próxima de zero e com pequena variância. Nas Redes Neurais os erros também tinham uma tendência da média em torno do zero, porém com uma variância visivelmente maior.

7.3 Comparação da Representatividade dos Modelos

Um desvio de 10% é amplamente aceitável para a modelagem aerodinâmica (RAJKUMAR; BARDINA, 2002). Para determinar-se a quantidade mínima disponível para modelagem dos coeficientes, foi feita uma análise do erro absoluto com a diminuição da massa de dados disponíveis. Esse resultado é importante para vericar qual método melhor modela um coeficiente específico o e qual a sua capacidade de predição frente a escassez de dados disponíveis.

Então, para cada coeciente aerodinâmico, foram gerados novos subconjuntos de treino igualmente espaçados com 90%, 80%, 70%, ..., 20% do conjunto de dados disponível, ou seja, os subconjuntos de treino tinham entre 500 e 2000 elementos. Além disso, a escolha dos pontos de treino é feita de forma aleatória, favorecendo a presença de imprecisões no modelo para testar a capacidade de inferência de cada método.

Decolagem Alfa

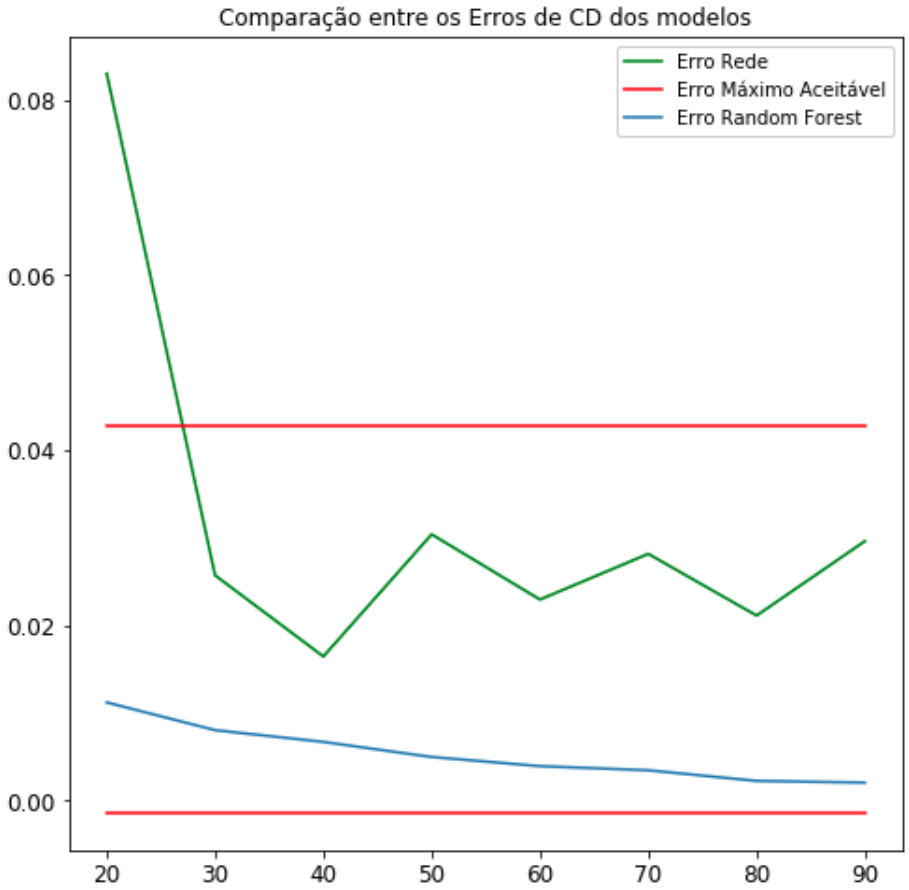


FIGURA 7.25 – Comparação entre os erros absolutos de C_d dos modelos e o erro máximo aceitável requerido para um simulador representativo.

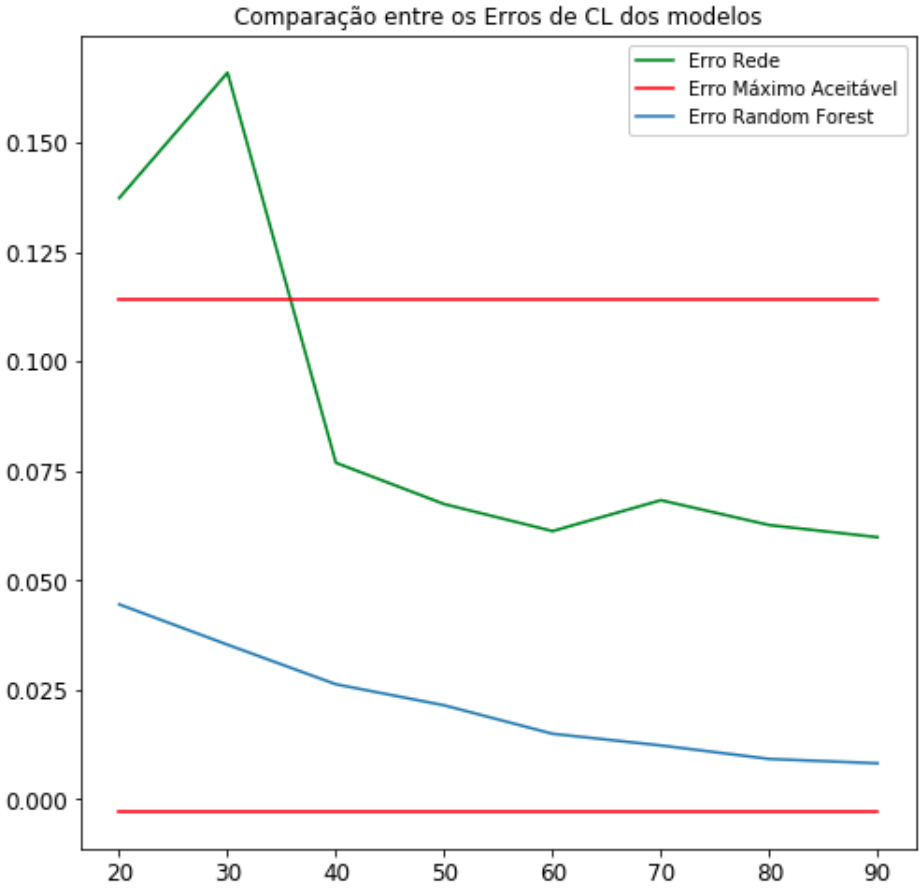


FIGURA 7.26 – Comparação entre os erros absolutos de C_l dos modelos e o erro máximo aceitável requerido para um simulador representativo.

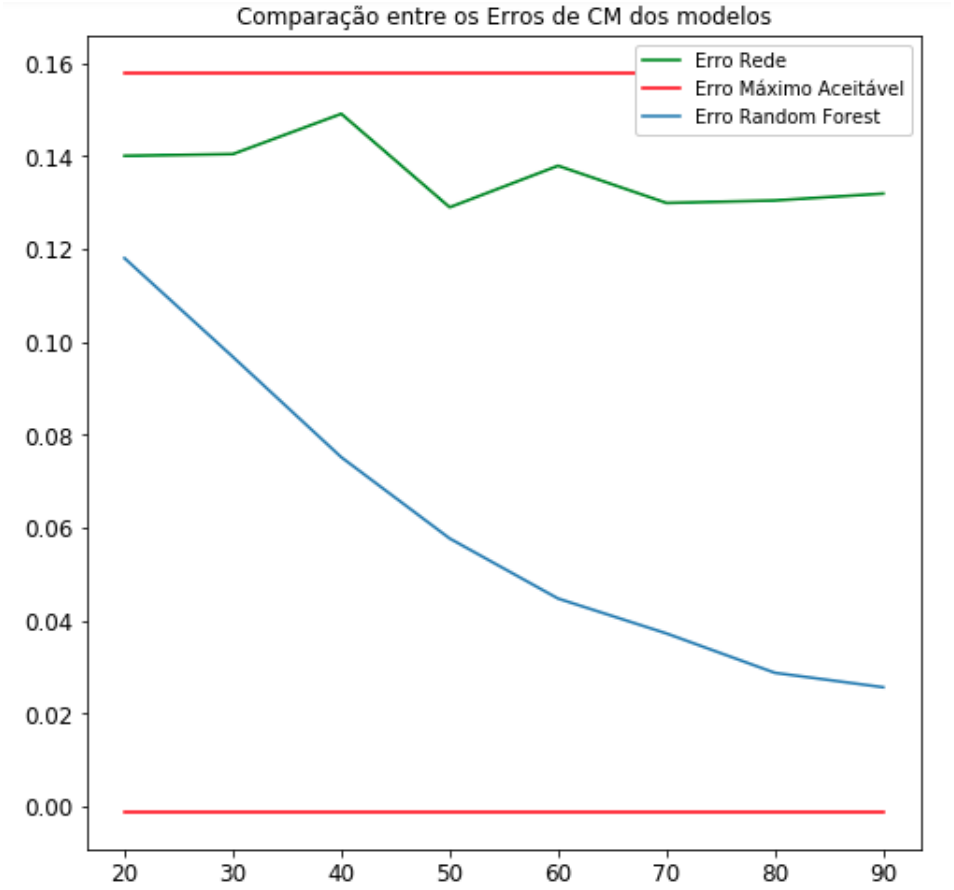


FIGURA 7.27 – Comparação entre os erros absolutos de C_m dos modelos e o erro máximo aceitável requerido para um simulador representativo.

Recolhido Alfa

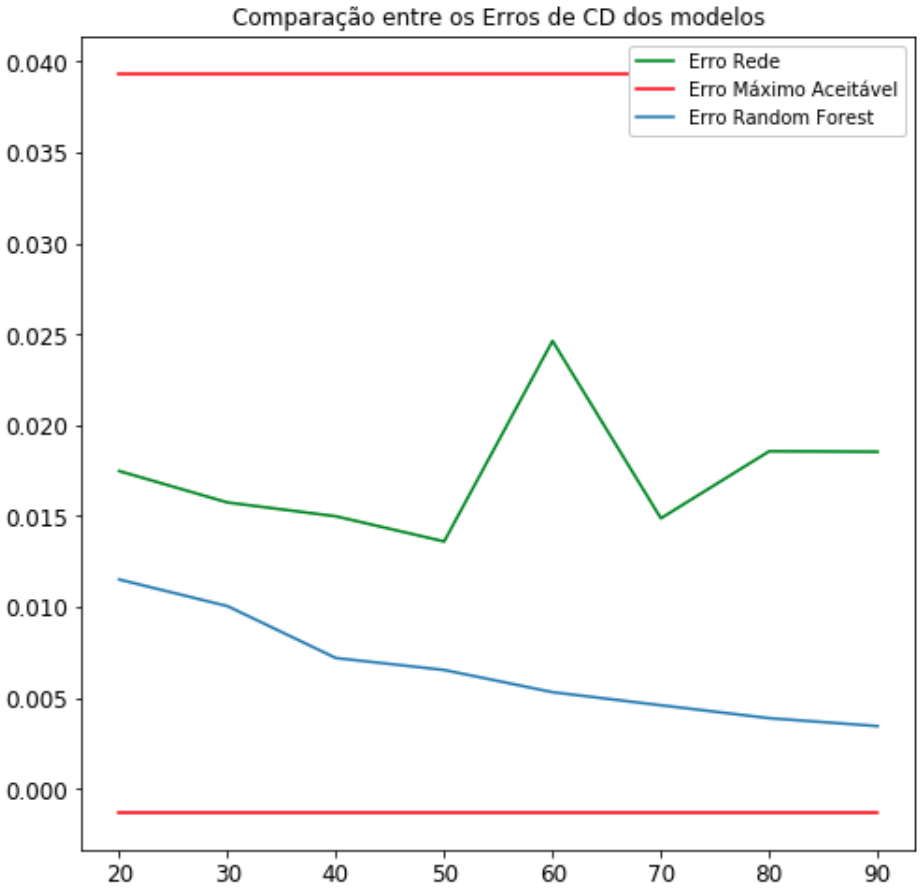


FIGURA 7.28 – Comparação entre os erros absolutos de C_d dos modelos e o erro máximo aceitável requerido para um simulador representativo.

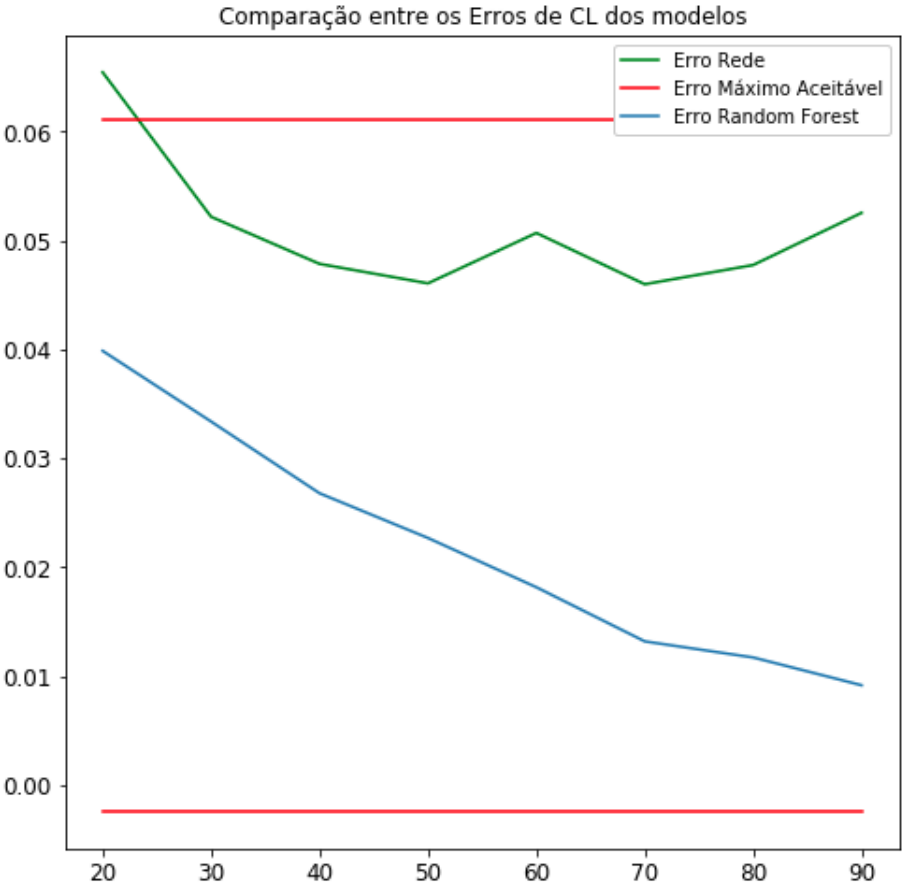


FIGURA 7.29 – Comparação entre os erros absolutos de C_l dos modelos e o erro máximo aceitável requerido para um simulador representativo.

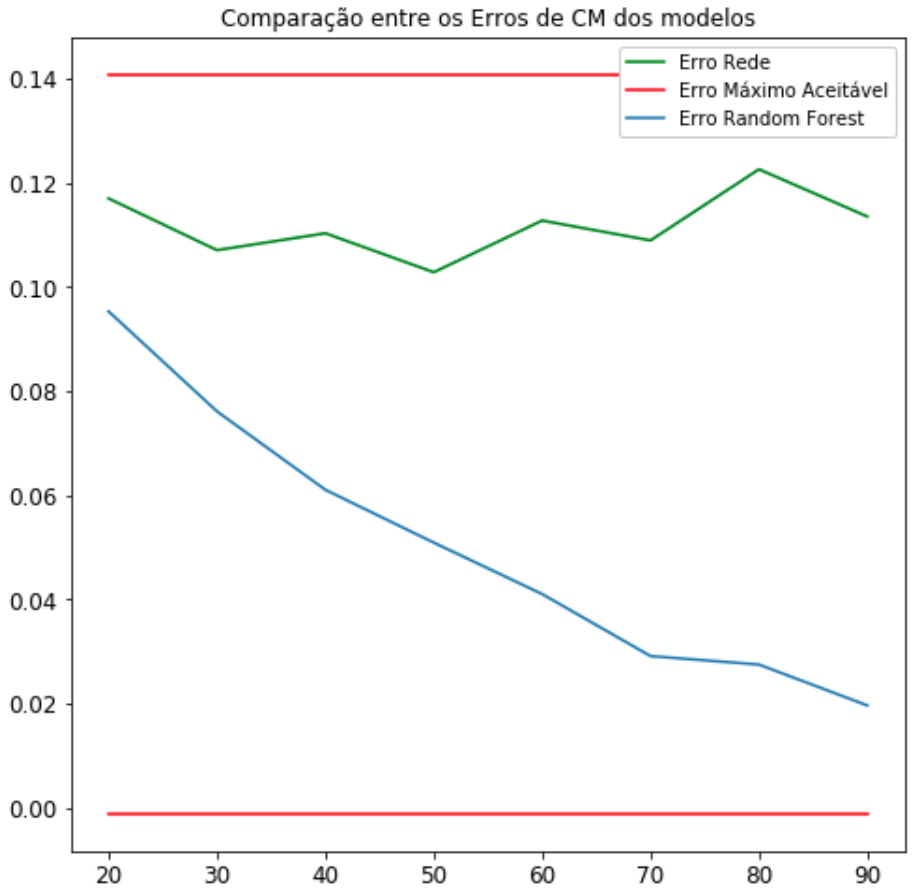


FIGURA 7.30 – Comparação entre os erros absolutos de C_m dos modelos e o erro máximo aceitável requerido para um simulador representativo.

Aterragem Alfa

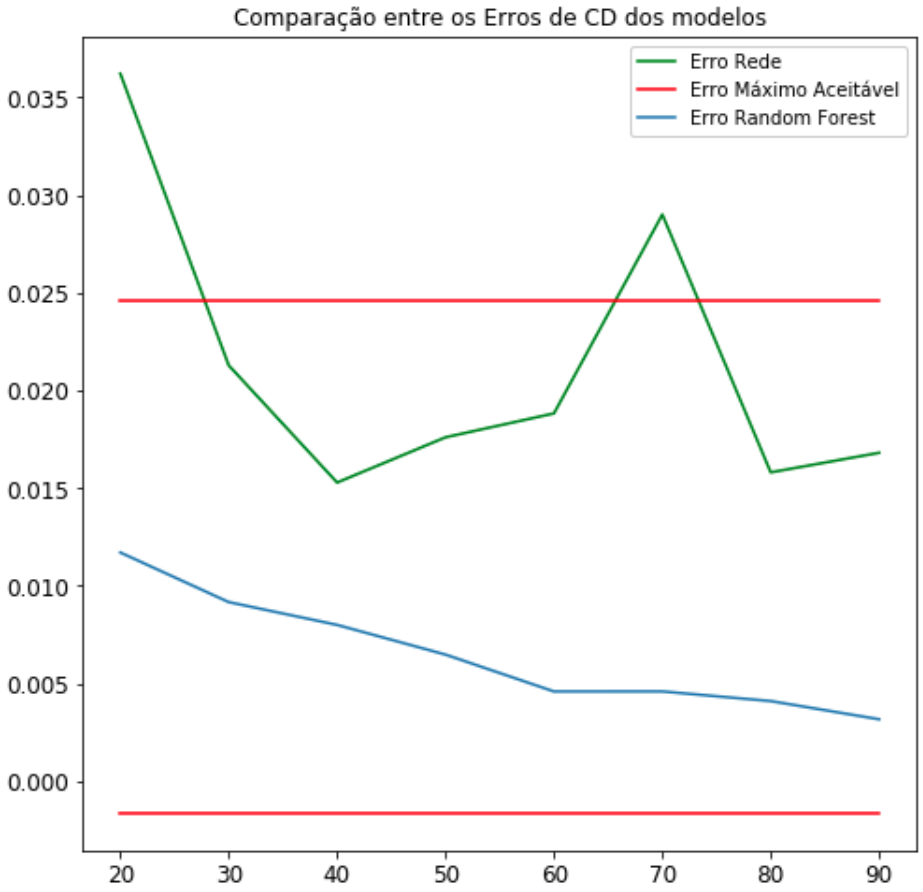


FIGURA 7.31 – Comparação entre os erros absolutos de C_d dos modelos e o erro máximo aceitável requerido para um simulador representativo.

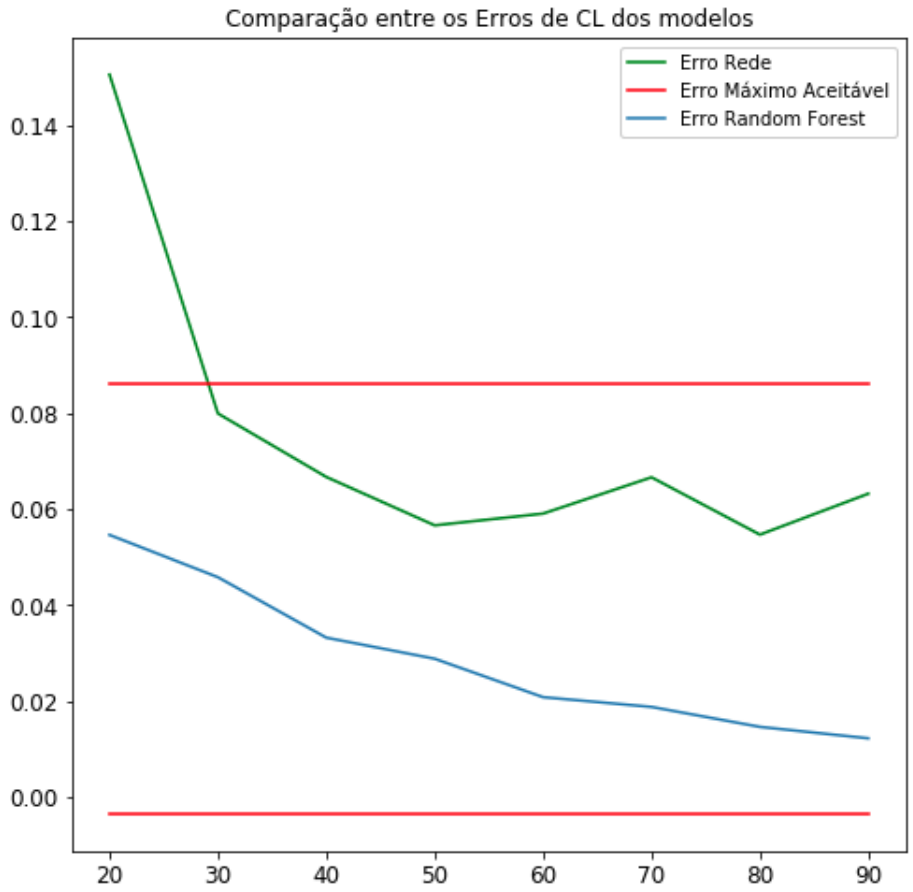


FIGURA 7.32 – Comparação entre os erros absolutos de C_l dos modelos e o erro máximo aceitável requerido para um simulador representativo.

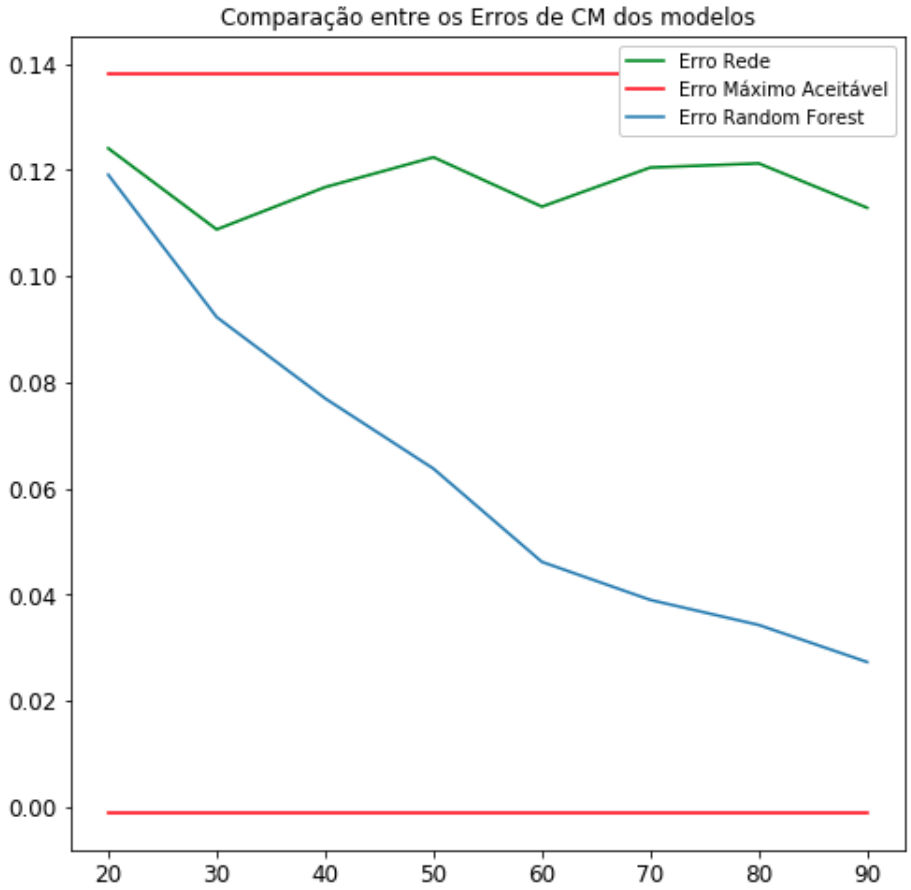


FIGURA 7.33 – Comparação entre os erros absolutos de C_m dos modelos e o erro máximo aceitável requerido para um simulador representativo.

Momentos

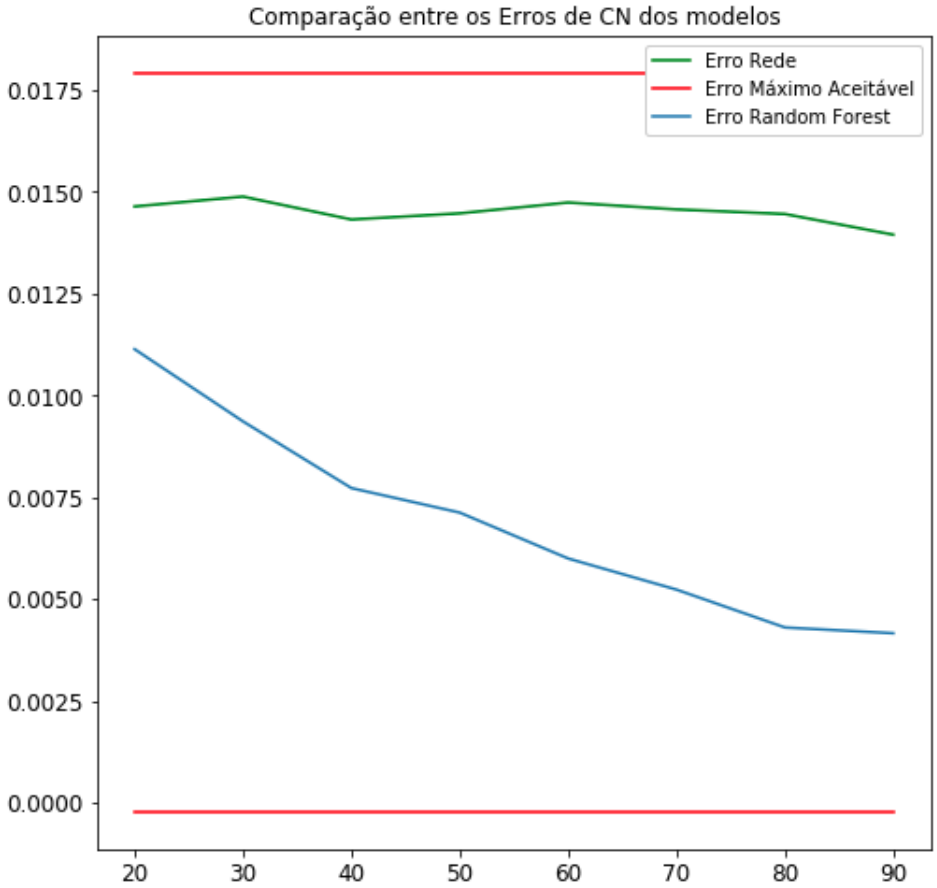


FIGURA 7.34 – Comparação entre os erros absolutos de C_n dos modelos e o erro máximo aceitável requerido para um simulador representativo.

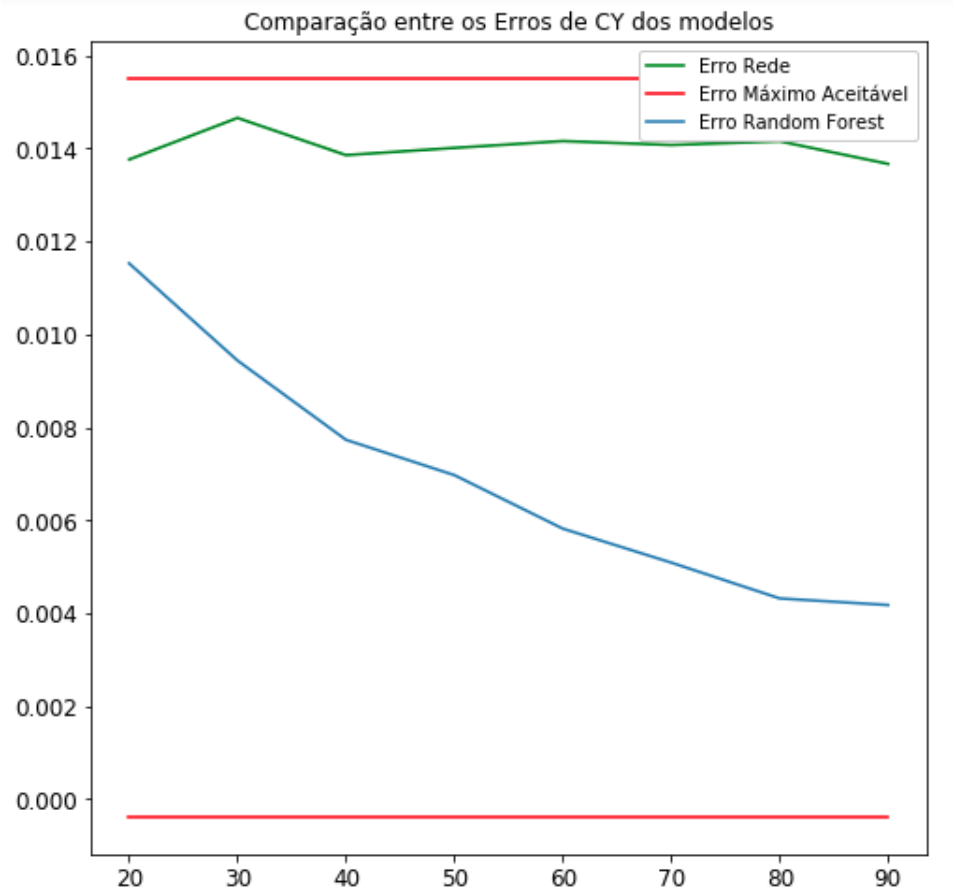


FIGURA 7.35 – Comparação entre os erros absolutos de C_y dos modelos e o erro máximo aceitável requerido para um simulador representativo.

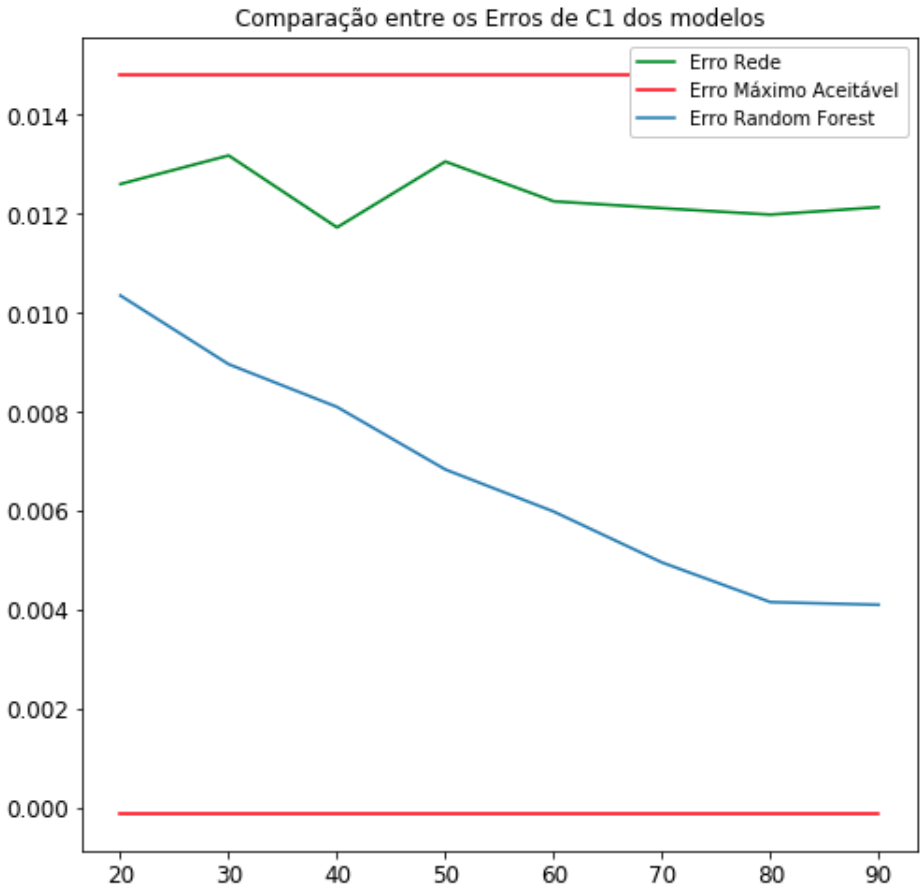


FIGURA 7.36 – Comparação entre os erros absolutos de C_1 dos modelos e o erro máximo aceitável requerido para um simulador representativo.

Dos gráficos anteriores, fica clara a vantagem dos modelos baseados em *Random Forest*. Mesmo com um conjunto de treino muito pequeno, é possível modelar de forma representativa a maioria dos coeficientes. Entretanto, ainda há variância nos dados. O modelo de Rede Neural apresentou alguns valores muito dispersos. A curva de diminuição dos erros com o aumento da quantidade de dados disponível para treino apresentou-se muito mais estável para a *Random Forest*, o que, complementando com o fato dela ter menor erro quadrático em relação à Rede com a quantidade total de treino, faz dela um modelo representativo dos coeficientes.

7.4 Comparação do Ganho Computacional dos Modelos

Agora, a avaliação e os resultados serão referentes ao tempo de execução, que possui uma característica importante para todos os simuladores durante os cálculos em tempo real. Para isso, utilizou-se uma biblioteca *open source* denominada PyFME (*Python Flight*

Mechanics Engine), que conta com todas as entidades necessárias para realização de uma simulação da cinemática de uma aeronave dadas as condições iniciais, como descrito no Capítulo 3. O escopo completo de como ocorre a simulação foge aos objetivos do trabalho, mas de maneira sumaria, é necessário um processo de integração envolvendo o cálculo das equações rotacionais, taxas angulares, ângulos de Euler e equações translacionais que regem a cinemática da aeronave . Basicamente, para comparar o tempo de execução entre os três métodos, interpolação, Rede e Floresta, foram alteradas, em código, as funções que calculam as forças aerodinâmicas na aeronave, utilizando os modelos descritos com apenas 40% dos dados de treino disponíveis, porcentagem suficiente para obtenção de um modelo representativo como já visto.

Considerando que as diferenças entre as simulações são apenas devido ao método de cálculo dos coeficientes, é feita uma integração de 90 *time steps*, em que a aeronave é submetida a *inputs* gerados dentro da ferramenta. O código da simulação é dado a seguir:

```
1
2 import time
3
4 from pyfme.simulator import Simulation
5
6 #aircraft possui os metodos de determinacao dos coeficientes ...
      aerodinamicos
7 #system
8 sim1 = Simulation(aircraft, system, environment, controls)
9
10 t0= time.clock()
11 results = sim1.propagate(90)
12 t1 = time.clock()
13 print("Time elapsed: ", t1 - t0) # CPU seconds elapsed (floating ...
      point)
```

Listing 7.1 – Simulação com a ferramenta PyFME

As entidades que são passadas para o construtor da classe Simulation foram definidas e a correta descrição delas foge ao objetivo do trabalho. Com a diferença $t_1 - t_0$, obtemos o tempo de execução da simulação. Portanto, para cada modelo, foram realizadas 100 simulações e o tempo médio obtido é dado a seguir:

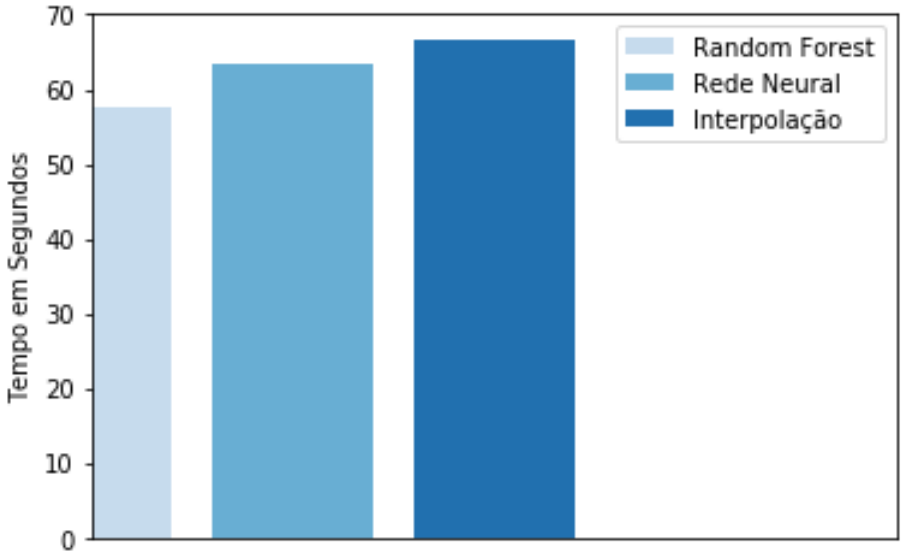


FIGURA 7.37 – Comparação entre os tempos médios de execução de simulação com os três métodos

Então , tem-se :

1. Random Forest : 58.5 s
2. Rede Neural : 63.5 s
3. Interpolação : 66.7 s

Portanto, o algoritmo em floresta mostra-se mais eficiente que os demais. Provavelmente, por tratar-se uma estrutura em árvore, poucos cálculos complexos são necessários, diferentemente de uma Rede Neural, que envolve desde produtos internos até cálculos de funções de ativação. Ambos mostraram-se superiores ao método clássico de interpolação, evidenciando que métodos modernos para avaliação de dinâmicas de voo devem ser estudados com maior profundidade.

8 Conclusões e Trabalhos Futuros

O presente trabalho apresentou um projeto completo de *Machine Learning*, desde a aquisição de *raw data*, dos ensaios em túnel de vento, até o treinamento e validação de um modelo representativo dos coeficientes aerodinâmicos. O método OCR, quando aplicado à imagens pré-processadas, apresentou uma leitura muito mais precisa e verídica. Da análise dos gráficos, verificou-se o correto comportamento dos coeficientes em função das entradas α e β. Da análise estatística reduziu-se os dados representando C_N, C_Y e C_1 para um único *Dataset*. Do treinamento, verificou-se a superioridade da *Random Forest* em relação à Rede, tanto na predição com menor conjunto de dados quanto na rapidez em que se calcula os coeficientes em ambiente simulado, por meio da ferramenta PyFME. Para trabalhos futuros, é sugerido a verificação dos modelos treinados no presente trabalho em ambientes gráficos, como o *Flight Simulator*.

Referências

ANDERSON, J. D. **Fundamentals of aerodynamics**. [S.l.]: McGraw-Hill Education, 2001.

HAN, J.; KAMBER, M.; PEI, J. **Data mining: concepts and techniques**. [S.l.]: Morgan Kaufmann, 2012.

SAMPAIO, O. S. **Ensaio em Túnel de Vento - EMB-312 TUCANO**: Primeira campanha de ensaio. [S.l.], 1985.

STEVENS, B. L.; LEWIS, F. L.; JOHNSON, E. N. **Aircraft control and simulation: dynamics, controls design, and autonomous systems**. [S.l.]: John Wiley e Sons, 2016.

FOLHA DE REGISTRO DO DOCUMENTO

1. CLASSIFICAÇÃO/TIPO	2. DATA	3. REGISTRO Nº	4. Nº DE PÁGINAS
TC	14 de novembro de 2019	DCTA/ITA/TC-081/2019	120

5. TÍTULO E SUBTÍTULO:

Determinação de coeficientes aerodinâmicos usando machine learning em python.

6. AUTOR(ES):

Bruno de Souza Neves

7. INSTITUIÇÃO(ÕES)/ÓRGÃO(S) INTERNO(S)/DIVISÃO(ÕES):

Instituto Tecnológico de Aeronáutica – ITA

8. PALAVRAS-CHAVE SUGERIDAS PELO AUTOR:

Coeficientes aerodinâmicos; Predição; Simulador; Regressão; Redes neurais; Random Forests;TensorFLow 2.0

9.PALAVRAS-CHAVE RESULTANTES DE INDEXAÇÃO:

Coeficientes aerodinâmicos; Predição; Redes neurais; Simulação de voo; Computação.

10. APRESENTAÇÃO: **X Nacional** **Internacional**

ITA, São José dos Campos. Curso de Graduação em Engenharia de Computação. Orientado: Prof. Dr. Paulo Marcelo Tasinaffo; coorientador: Ten Cel Piterson Lisboa. Publicado em 2019.

11. RESUMO:

O trabalho propõe-se a desenvolver uma metodologia em linguagem Python capaz de estimar de forma confiável e rápida os coeficientes aerodinâmicos da aeronave EMB-312 TUCANO para um simulador de voo com dados dispersos. As abordagens principais para a resolução do problema serão obtidas utilizando Redes Neurais e Random Forests incluídos nas tecnologias Tensor Flow e Keras. Os dados de treinamento para as redes são derivados de ensaios em túnel de vento e os coeficientes são modelados como funções adimensionais de características do escoamento, pressão dinâmica, e das deflexões superfícies de controle da aeronave. Esse estudo tem o intuito de diminuir a quantidade de pontos tomados em uma campanha de ensaio em túnel, diminuindo os custos operacionais para desenvolvimento de um modelo aerodinâmico preciso. Será realizada um estudo de teoria aerodinâmica análise exploratória dos dados (análise gráfica), para detecção de possíveis ruídos e correlação entre os dados e preparação e tratamento dos dados para serem fornecidos aos algoritmos de treinamento. Então, é feita uma análise comparativa entre os diversos algoritmos de aprendizagem de máquina, avaliando-se a redução da quantidade de dados em função do erro final.

12. GRAU DE SIGILO:

(X) OSTENSIVO **( ) RESERVADO** **( ) SECRETO**