Uma Nova Abordagem de Machine Learning para a Previsão Espaço-Temporal da Velocidade do Vento

Bruno Q. Bastos
Departamento de Engenharia Industrial
PUC-Rio
Rio de Janeiro, Brasil
brunoq.b@aluno.puc-rio.br

Fernando L. Cyrino Oliveira
Departamento de Engenharia Industrial
PUC-Rio
Rio de Janeiro, Brasil

***Resumo*— O crescimento da penetração de energia eólica em sistemas elétricos de potência traz desafios para a operação dos sistemas elétricos modernos. Neste contexto, previsões acuradas da velocidade do vento são de grande importância, pois ajudariam a reduzir a incerteza relacionada ao vento, potencialmente reduzindo custos operacionais. Nosso trabalho propõe uma nova abordagem ao problema de previsão espaço-temporal da velocidade do vento, construindo modelos que aprendem a relacionar variáveis meteorológicas e coordenadas geográficas com a velocidade do vento. Estes modelos são utilizamos para prever o vento em diferentes localidades. Utilizamos comitês de árvores de decisão como o modelo de *machine learning* para resolver o problema proposto. Inicialmente, testamos diferentes árvores de decisão, e escolhemos a configuração que leva aos melhores resultados. Em seguida, utilizamos a mesma configuração para treinar os comitês, com algoritmos Adaboost e *Random Forests*. Nossos resultados mostram que a *Random Forest* apresenta os melhores resultados.**

***Index Terms*—Árvores de Decisão, Machine Learning, Vento, Previsão.**

***Abstract*— The increasing penetration of wind energy in power systems introduces new challenges in reliable operation of modern power systems. In this context, accurate wind speed prediction is of great importance, because help reduce uncertainty related to wind resources, potentially reducing operational costs in power systems. Our work proposes a new approach to the task of spatio-temporal wind speed prediction, constructing models which learn to relate meteorological variables and geographical coordinates with wind speed. These models are used to predict wind speed at different locations within a region. We use ensemble of decision trees as the machine learning model to solve the proposed task. We test different decision trees and choose the configuration which leads to the best results. Following this, we use the best configuration to train the ensemble of decision trees, with Adaboost and Random Forest algorithms. Our results show that the Random Forest algorithm presents the best results.**

***Index Terms*—Decision Trees, Machine Learning, Wind, Forecasting.**

https://proceedings.science/p/85339?lang=pt-br

I. INTRODUÇÃO

O aumento da geração de energia eólica em sistemas de potência tem tornado cada vez mais relevante o estudo da variabilidade e da previsão do vento, sendo estes tópicos importantes tanto para o planejamento da operação, como o da manutenção e da expansão de sistemas elétricos. O aumento do uso da fonte eólica pode fazer com que se necessite maior reserva operacional [1] e melhor balanceamento da geração e demanda de curtíssimo prazo, possivelmente ocasionando aumento em custos operacionais. Adicionalmente, o sistema de transmissão deveria ser suficiente para permitir importação e exportação de potência entre regiões, quando a potência eólica gerada é baixa ou alta. Neste sentido, entender o fenômeno do vento é essencial para responder muitas questões do sistema elétrico.

Em específico, a produção de previsões acuradas da velocidade do vento e da potência eólica pode fazer com que o problema do balanceamento do sistema de potência seja menos afetado pela variabilidade do vento, conforme indicado em [2]. Muitos estudos abordaram o problema da previsão da velocidade e da potência do vento, utilizando diferentes abordagens para a previsão [3]. Os modelos utilizados para solucionar o problema podem ser classificados em quatro categorias amplas, conforme sugerido em [4]: modelos físicos, modelos estatísticos convencionais, modelos de correlação espacial, e modelos e inteligência artificial.

Modelos físicos utilizam métodos numéricos para estimar o estado da atmosfera, considerando as leis que descrevem o fenômeno atmosférico [5], [6]. De acordo com [7], este tipo de modelo não é adequado para a previsão de curto-prazo. Modelos estatísticos convencionais utilizam dados históricos da velocidade do vento para prever a velocidade do vento no futuro. Nesta classe de modelos, os modelos clássicos de séries temporais, como os propostos por Box & Jenkins [8] – i.e., modelo autoregressivo (AR), modelo de média móvel (MA), modelo autoregressivo de média móvel (ARMA) e modelo autoregressivo integrado de média móvel (ARIMA) – são comumente utilizados [3]. De acordo com [3], os modelos estatísticos usualmente apresentam bom desempenho na previsão de curto-, médio- e longo-prazo da velocidade do vento. Modelos de correlação espacial, por sua vez, utilizam informação de sítios vizinhos para estimar a velocidade do vento em determinada localidade, onde as medições do vento não estão disponíveis. Mais recentemente, com o desenvolvimento e popularização de técnicas de inteligência artificial, modelos de inteligência artificial, como *support vector regression* (SVR) [6], redes neurais artificiais (ANNs) [9], sistemas de inferência fuzzy [10], e muitos outros, têm sido adotados para prever a velocidade do vento [6]. Para uma revisão dos modelos utilizados para prever o vento, consulte os artigos [4], [7], [11] e as referências neles contidas.

Este trabalho tem como objetivo construir um modelo de aprendizado de máquinas que produza previsões um passo à frente no tempo para a velocidade do vento em uma determinada região do Brasil. O principal objetivo é o de construir um modelo que relacione dados meteorológicos (e.g., temperatura, pressão no nível do mar, humidade relativa) – no tempo presente e passado – e coordenadas geográficas (i.e., latitude e longitude da região estudada) com a velocidade do vento uma hora à frente. Neste sentido, a abordagem proposta é a de modelagem espaço-temporal, uma vez que nosso modelo de aprendizado de máquinas é treinado com informações espaciais e temporais.

A base de dados utilizada no trabalho é aquela do *Climate Forecast System Reanalysis* (CFSR), produzida e atualizada pelo *National Centers for Environmental Prediction* (NCEP), uma organização governamental americana.[1] O dataset CFSR contém dados de séries temporais horárias para um conjunto de variáveis, tais como temperatura, umidade, precipitação, velocidade do vento, entre outros, disponíveis em diferentes resoluções espaciais. Estas séries horárias são produzidas por um modelo físico, o *Climate Forecast System* (CFS). A principal motivação para utilizar o CFSR provém do fato de ser um repositório público com muitas informações disponíveis, que nos permitem investigar modelos espaço-temporais para previsão do vento. Apesar de ser um dataset de reanálise, baseado em simulação atmosférica, é válido para nossos propósitos investigativos. Os dados utilizados do CFSR estão relacionados ao estado da Bahia, no Nordeste do Brasil – onde a capacidade eólica é mais abundante –, e são descritos na Seção III deste trabalho.

[1] A base de dados pode ser obtida no website https://rda.ucar.edu/.

https://proceedings.science/p/85339?lang=pt-br

Neste artigo, propomos o uso de comitês de árvores de decisão (*decision trees*, DTs) para resolver o problema da previsão uma hora à frente da velocidade do vento, tendo como base dados meteorológicos e coordenadas geográficas. De acordo com [12], as árvores de decisão são um dos algoritmos mais popularmente adotados para a prática de inferência indutiva, tendo sido utilizadas com sucesso na resolução de muitas tarefas. As DTs têm muitas vantagens, como, por exemplo: (1) são rápidas de se construir, (2) oferecem interpretabilidade, (3) internamente realizam seleção de atributos, (4) incorporam valores numéricos e categóricos, e outros [13]. Nossa proposta de utilizar comitês de árvores de decisão é justificada pelo fato de que técnicas de agrupamento de previsores tendem a melhorar o desempenho final da previsão (e.g., ver o exemplo de *boosting* de DTs em [13]). Neste contexto, nós aplicamos as técnicas de AdaBoost [14] e *Random Forests* [15] para criar comitês de DTs, com o objetivo de criar melhores previsores para nossa tarefa.

A organização deste artigo é a seguinte: na Seção II, é apresentada a tarefa a qual nosso trabalho resolve, i.e., uma tarefa de aprendizado supervisionado, onde desenvolvemos um regressor que relaciona variáveis meteorológicas com o vento. Logo depois, na Seção III, nós descrevemos a base de dados utilizada no trabalho: fonte de referência, atributos obtidos, volumetria, etc. Na Seção IV, apresentamos os experimentos realizados para resolver a tarefa proposta: análise de desempenho de DTs para diferentes profundidades, seguido de aplicação de comitês de DTs para a melhor profundidade obtida. Em seguida, na Seção V, são apresentados nossos resultados, e, na Seção VI, apresentamos nossas principais conclusões e ideias de trabalhos futuros.

II. TAREFA

A tarefa deste trabalho é aquela de prever a velocidade do vento, uma hora à frente do tempo, para qualquer localização (coordenada geográfica) dentro de uma região, utilizando apenas informações meteorológicas. Neste sentido, a ideia é construir um modelo $f: \mathbf{x} \rightarrow y$, que relacione informações de entrada, $\mathbf{x}$, de temperatura, umidade, pressão, precipitação e chuva nos instantes presentes e passados, e de coordenadas geográficas, com informações de saída, y – a velocidade do vento uma hora à frente do tempo. Este arranjo da tarefa está ilustrado na Figura 1.

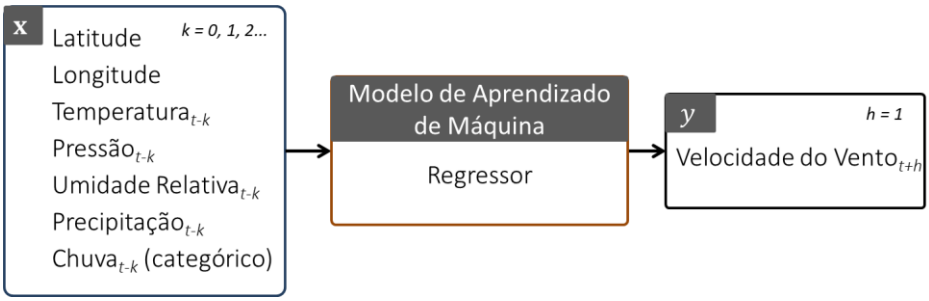


Figura 1 – Framework da tarefa

Os modelos investigados são treinados com informação espacial; isto é, a latitude e longitude são entradas do modelo. Além disso, os exemplos fornecidos aos modelos (i.e., os exemplos com os quais os modelos são treinados) são dados para coordenadas únicas (i.e., as informações contidas em cada exemplo dizem respeito a apenas uma coordenada, sem relação explícita com outras coordenadas). Desta maneira, os modelos são treinados com informações de diferentes sítios. Num âmbito mais amplo, portanto, o modelo utiliza informações de diferentes localidades para aprender a relação inerente entre meteorologia e coordenadas (i.e., a entrada do modelo, $\mathbf{x}$) e a velocidade do vento um passo à frente (i.e., a saída do modelo, y). Este tipo de abordagem é bastante interessante, porque o modelo poderia vir a prever a velocidade do vento para locais onde dados não estão disponíveis (mas onde dados meteorológicos estão disponíveis).

III. BASE DE DADOS

A base de dados utilizada neste trabalho foi a CFSR; ela foi obtida no website do Research Data Archive (RDA) (ver nota de rodapé de número 1). Nós extraímos séries temporais de frequência horária para as seguintes variáveis: (i) chuva (categórico), (ii) precipitação, (iii) umidade relativa, (iv) temperatura, (v) pressão no nível do mar, (vi)

https://proceedings.science/p/85339?lang=pt-br

componente u do vento, e (vii) componente v do vento. As séries temporais destas variáveis foram obtidas para pares de coordenada dentro da área sombreada ilustrada na Figura 2. Os valores de latitude variam de -19° a -7°, e os valores de longitude variam de -47° a -37°. Tanto os valores de latitude como os de longitude variam de 0.5° em 0.5° (ou seja, a resolução espacial dos nossos dados é de 0.5° x 0.5°). A região escolhida para o estudo foi aquela que envolve completamente o estado da Bahia, na Região Nordeste,, e um pedaço do estado de Minas Gerais. As séries temporais foram obtidas para os tempos de 2011-04-01 00:00 a 2017-01-01 00:00., com uma resolução temporal horária.



Figura 2 – Região contemplada no estudo (imagem obtida do Google Maps).

As Figuras 3 e 4 ilustram os histogramas das variáveis listadas acima, para uma única coordenada (latitude -12.5° e longitude -41.5°) e para todas as coordenadas contidas na região sombreada, respectivamente. É possível notar que as distribuições das variáveis mudam sobremaneira, quando consideramos todas as coordenadas da região. Por exemplo, os valores dos componentes u e v do vento ganham maior amplitude; a pressão no nível do mar passa de uma distribuição que se assemelha a uma forma triangular para uma distribuição que se assemelha a uma distribuição normal; a distribuição da umidade relativa, por sua vez, ganha um pico entre os valores 0 e 1. Estes fatores demonstram que, dependendo da coordenada onde se pretende prever a velocidade do vento (e também do tempo) os valores das variáveis se modificam, motivo pelo qual, incluir informações espaciais na previsão pode ser benéfico.

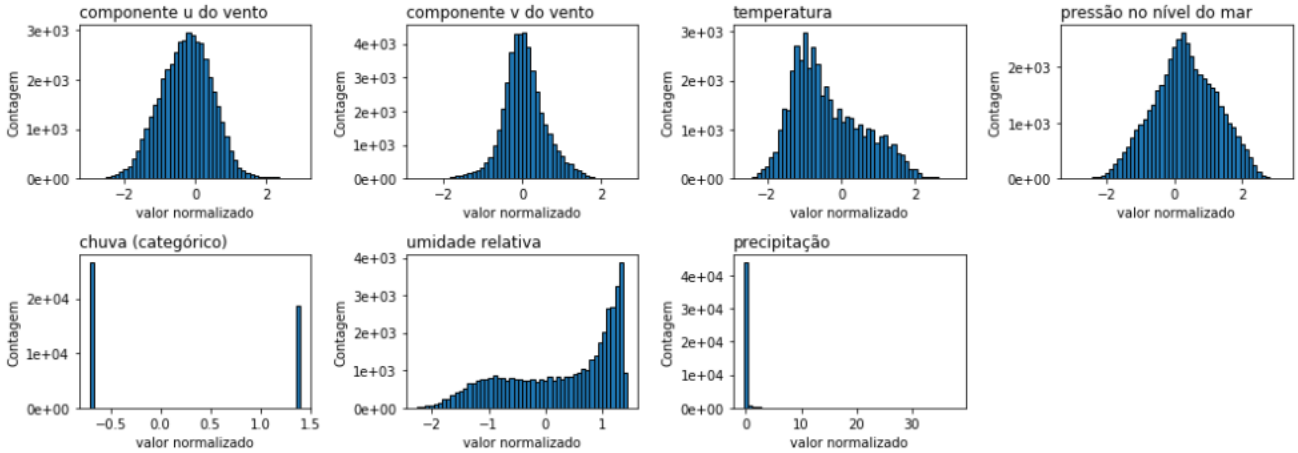


Figura 3 – Histogramas das variáveis explicativas (normalizadas) para a coordenada -12.5 de latitude e -41.5 de longitude

https://proceedings.science/p/85339?lang=pt-br

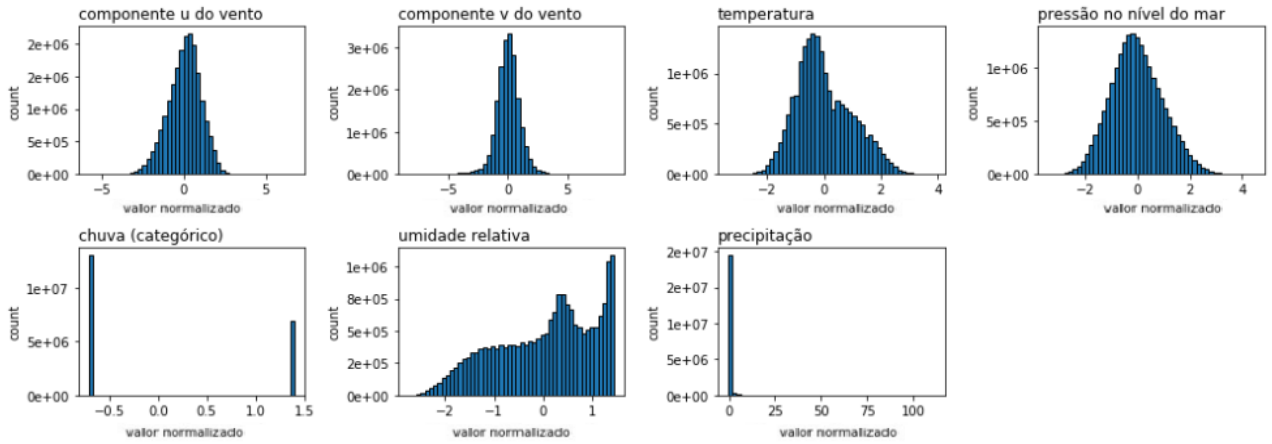


Figura 4 – Histogramas das variáveis explicativas (normalizadas) para todas as coordenadas

Os dados extraídos do RDA foram baixados em formato netCDF. Cada uma das variáveis tinha dados distribuídos em um conjunto de 69 documentos, um para cada mês de estudo. Ao final, a base de dados totalizava 414 documentos. Nós utilizamos o pacote netCDF4[2] para ler estes documentos para a linguagem Python, e utilizamos os pacotes pandas[3] e numpy[4] para manipular os dados e criar uma matriz onde as linhas eram exemplos, e as colunas, atributos, e um vetor com os *targets* (alvos) da tarefa. Após a manipulação dos dados, obtivemos uma matriz com 26,483,100 linhas e 17 colunas – contendo exemplos, $\mathbf{x}$ – e um vetor de tamanho 26,483,100 – contendo os *targets*, y.

A variável de interesse é a velocidade do vento no tempo $t + 1$, e os 17 atributos eram os seguintes:

- (i)-(iii) Chuva (variável categórica) nos tempos t, $t - 1$ and $t - 2$
- (iv)-(vi) Precipitação total nos tempos t, $t - 1$ and $t - 2$
- (vii)-(ix) Pressão no nível do mar nos tempos t, $t - 1$ and $t - 2$
- (x)-(xii) Umidade relativa nos tempos t, $t - 1$ and $t - 2$
- (xiii)-(xv) Temperatura nos tempos t, $t - 1$ and $t - 2$
- (xvi) Latitude
- (xvii) Longitude

Para que pudéssemos avaliar o desempenho dos modelos na tarefa proposta, dividimos o dataset em três conjuntos: treinamento, validação e teste. O conjunto de treinamento foi utilizado para treinar os modelos, i.e., para encontrar os parâmetros do modelo que otimizam uma determinada função de custo do problema. O conjunto de validação foi utilizado para avaliar o desempenho do modelo na previsão de dados com os quais ele não foi treinado. Conforme os resultados são obtidos, os hiperparâmetros do modelo podem ser modificados, e novos modelos podem ser treinados. Após se obter o melhor resultado possível, i.e., aquele com melhor acurácia e generalização, o conjunto de teste é utilizado para avaliar, por fim, como o modelo desempenha em dados nunca vistos antes. O dataset de treino continha informações para todas as coordenadas, excetuando-se as seguintes coordenadas: (-10.5°, -41.5°), (-10.5°, -44°), (-15°, -41.5°), (-15°, -44°), que foram coordenadas centrais, equidistantes das laterais da área sombreada na Figura 2. Os dadasets de validação e teste continham, cada, exemplos das quatro coordenadas (que foram escondidas no treino). Após esta manipulação, estávamos prontos para rodar nossos modelos. Na seção seguinte, descrevemos os experimentos do trabalho.

[2] http://unidata.github.io/netcdf4-python/
[3] http://pandas.pydata.org/
[4] http://www.numpy.org/

https://proceedings.science/p/85339?lang=pt-br

IV. EXPERIMENTOS

Os experimentos com comitês de DTs foram implementados na linguagem Python, com o pacote scikit-learn [16]. Primeiro, iniciamos os experimentos testando diferentes DTs para o problema, variando a profundidade das árvores (para uma descrição sobre DTs, por favor ver [12]). Testamos DTs com profundidades máximas de 3 a 22. Abaixo, algumas especificações das árvores de decisão que construímos. (Utilizamos a função *sklearn.tree.DecisionTreeRegressor* do pacote scikit-learn).

- Profundidade: 3 a 22
- Critério de qualidade da divisão dos nós da árvore: erro médio absoluto
- Estratégia de divisão: melhor divisão de acordo com o critério escolhido
- Mínimo número de amostras para se dividir um nó: 2
- Mínimo número de amostras para se ter num nó: 1

Seguindo a implementação das 21 DTs, nós avaliamos as métricas de acurácia (descritas no final desta seção) nos conjuntos de treinamento e de validação. Verificamos qual configuração (i.e., máxima profundidade) da DT produz a melhor acurácia. Com isso, determinamos o valor da máxima profundidade para utilizar nos algoritmos de comitê de DTs: Adaboost [14] de DTs e Random Forests (RFs) [15]. Com esta abordagem, os resultados dos comitês seriam, no pior dos casos, muito próximos aos da melhor DT que encontramos. Em seguida, testamos três configurações de Adaboost (implementadas com as funções *sklearn.ensemble.AdaBoostRegressor* e *sklearn.tree.DecisionTreeRegressor* do scikit-learn) e duas configurações de RFs (implementadas com a função *sklearn.ensemble.RandomForestRegressor*¶ do mesmo pacote Python), que detalhamos na seção de resultados. Nós comparamos a proposta técnica de comitês de DTs contra um modelo de regressão linear (implementado com a função *sklearn.linear_model.LinearRegression*¶ do scikit-learn).

As métricas de acurácia utilizadas para avaliar o desempenho dos modelos foram a raíz do erro quadrático médio (*root means square error* – RMSE) e o coeficiente de determinação (conhecido como R^2). Após investigarmos a literatura, notamos que o RMSE era amplamente utilizado como métrica de acurácia de tarefas de previsão da velocidade do vento (e.g., [1], [2]). Por este motivo, decidimos utilizar o RMSE como a principal métrica de acurácia do nosso trabalho. O coeficiente de determinação também foi medido, porque é um modo direto de enteder o quanto da tarefa o nosso modelo está "explicando".

Sejam $\hat{y}_i$ e y_i, respectivamente, o valor estimado e observado da velocidade do vento (em *m/s*) da amostra $i \in S$, onde S é o conjunto de amostras; seja n_s o número de amostras em S; e seja $\bar{y} = (1/n_s)\sum_{i \in S} y_i$, então a Eq. (1) e a Eq (2) detalham a formulação do RMSE e do R^2, respectively.

$$RMSE = \sqrt{\frac{\sum_{i \in S}(\hat{y}_i - y_i)^2}{n_S}} \quad (1)$$

$$R^2 = 1 - \frac{\sum_{i \in S}(\hat{y}_i - y_i)^2}{\sum_{i \in S}(\hat{y}_i - \bar{y})} \quad (2)$$

V. RESULTADOS

Após executar o código em Python, que constrói, treina e avalia as 21 DTs (ver Apêndice), obtivemos os resultados apresentados na Tabela I, para os conjuntos de treino e validação.

Tabela I – Resultados de R^2 e RMSE para todas as DTs simples

DTs	**R^2**		**RMSE (m/s)**	
	Treinamento	**Validação**	**Treinamento**	**Validação**
Profundidade 3	0.4673	0.1169	1.4046	1.3707
Profundidade 4	0.5147	0.2191	1.3406	1.2890
Profundidade 5	0.5574	0.3417	1.2803	1.1835

https://proceedings.science/p/85339?lang=pt-br

Profundidade 6	0.5925	0.3538	1.2284	1.1725
Profundidade 7	0.6163	0.3732	1.1921	1.1548
Profundidade 8	0.6370	0.4156	1.1595	1.1150
Profundidade 9	0.6523	0.4471	1.1348	1.0846
Profundidade 10	0.6656	0.4739	1.1128	1.0580
Profundidade 11	0.6777	0.4340	1.0925	1.0973
Profundidade 12	0.6885	0.4535	1.0741	1.0783
Profundidade 12	0.6885	0.4535	1.0741	1.0783
Profundidade 13	0.6989	0.4736	1.0561	1.0583
Profundidade 14	0.7086	0.4993	1.0389	1.0321
Profundidade 15	0.7178	0.5092	1.0223	1.0218
Profundidade 16	0.7272	0.5144	1.0051	1.0164
Profundidade 17	0.7371	0.5153	0.9868	1.0156
Profundidade 18	0.7475	0.5113	0.9670	1.0197
Profundidade 19	0.7589	0.5037	0.9449	1.0276
Profundidade 20	0.7715	0.4953	0.9199	1.0362
Profundidade 21	0.7854	0.4833	0.8914	1.0485
Profundidade 22	0.8007	0.4691	0.8591	1.0628

Dos resultados apresentados na Tabela I, verificamos que as maiores profundidades são aquelas que retornam melhores valores de R^2 e RMSE no conjunto de validação. Por este motivo, escolhemos o valor de máxima profundidade para o Adaboost de DTs e para as RFs como 20. O melhor valor de máxima profundidade, para uma DT simples é o 17. Entretanto, decidimos escolher um valor maior de profundidade, pois incluiria DTs de tamanho 17 na solução, de forma que o modelo tentaria encontrar configurações que produzissem resultados, ao menos, próximos do melhor resultado encontrado na Tabela I.

Nós, então, definimos este hiperparâmetro central nos comitês de DTs, e rodamos nossos códigos (ver Apêndice). A Tabela II mostra as configurações de comitês testados nesta fase do trabalho, e os resultados de R^2 e RMSE para os dados dos conjuntos de treinamento e validação. É possível ver que as RFs e o Adaboost de DTs apresentaram resultados bastante similares, com a RF de 10 DTs com máxima profundidade de 20 sendo o melhor modelo no conjunto de validação.

Tabela II – Resultados de R^2 and RMSE para os modelos de RF e Adaboost

Algorithm	**Nº of DTs**	**DTs max depth**	**R^2**		**RMSE (m/s)**	
			training	**validation**	**Training**	**validation**
RF	5	20	0.7820	0.5441	0.8986	0.9849
RF	***10***	***20***	***0.7859***	***0.5675***	***0.8905***	***0.9593***
Adaboost	5	20	0.7820	0.5464	0.8985	0.9824
Adaboost	10	20	0.7887	0.5639	0.8846	0.9632
Adaboost	20	5	0.5831	0.5105	1.2426	1.0205

Em seguida, com intuito de entender como os comitês de DTs se comparam com o modelo de regressão linear, rodamos, também, a regressão linear. Nas Figuras 5 e 6, mostramos os resultados de RMSE e R^2, respectivamente, para o modelo de regressão linear, para a melhor RF, melhor Adaboost e melhor DT – para todos os conjuntos de dados.

https://proceedings.science/p/85339?lang=pt-br

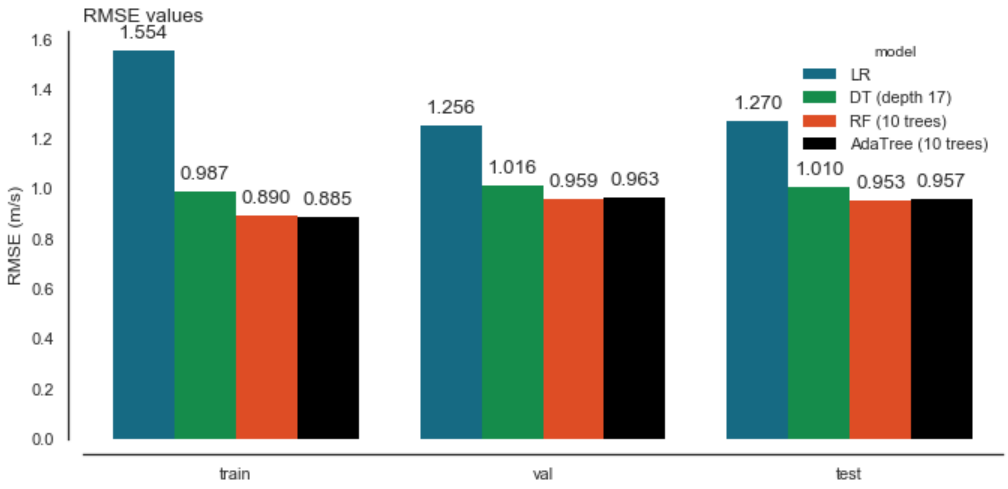


Figura 5 – Resultados de RMSE da regressão linear (LR) e das melhores DT, RF e Adaboost de DTs (AdaTree)

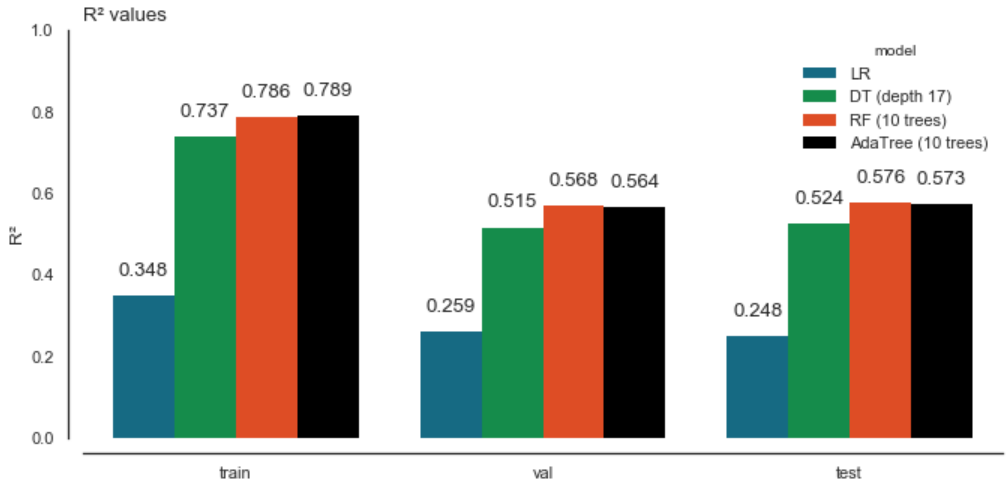


Figura 6 – Resultados de R^2 da regressão linear (LR) e das melhores DT, RF e Adaboost de DTs (AdaTree)

É possível notar que a abordagem de se utilizar comitês de DTs, para resolver a tarefa da previsão da velocidade do vento, supera o modelo baseline, mas se mostra apenas um pouco melhor que a DT simples, o que indica que há espaço para melhorias na modelagem. Ao comparar nossos resultados com os da literatura (e.g., RMSEs nos artigos [1] e [17]), notamos que nossos modelos estão apresentando erros na escala correta (não podemos, entretanto, tirar muitas conclusões contra estas referências, pois os datasets são distintos).

VI. CONCLUSÕES

O uso de comitês de árvores de decisão foi proposta para resolver a tarefa de previsão espaço-temporal da velocidade do vento uma hora à frente do tempo. Informações meteorológicas para diferentes localidades foram utilizadas para construir um modelo capaz de inferir a velocidade do vento para qualquer localidade (dentro da região de estudo), caso as informações meteorológicas sejam fornecidas como entrada ao modelo. O algoritmo de *Random Forests* com 10 árvores de decisão, de profundidade máxima de 20, apresentou o melhor resultado para a tarefa, apesar um pouco melhor que a Adaboost de DTs. As técnincas de comitê superaram, em termos de acurácia, o modelo de regressão linear, mas foram apenas um pouco melhores do que a árvore de decisão simples. Apesar deste resultado, a abordagem proposta para o problema da previsão espaço-temporal é inovadora, e promissora, propondo utilizar algoritmos de *machine learning* para aprender a relação entre variáveis meteorológicas,

https://proceedings.science/p/85339?lang=pt-br

coordenadas geográficas e velocidade do vento futura. Trabalhos futuros poderiam incluir, no modelo, atributos com informações diretas do sítios vizinhos (ao invés de deixá-los implícitos, nas amostras do dataset). Além disso, seria necessário avaliar se a abordagem proposta para a previsão espaço-temporal produz resultados melhores que o da abordagem naïve, com previsões para sítios únicos.

VII. REFERÊNCIAS BIBLIOGRÁFICAS

[1] D. C. Hill, D. McMillan, K. R. W. Bell, and D. Infield, "Application of auto-regressive models to U.K. wind speed data for power system impact studies," *IEEE Trans. Sustain. Energy*, vol. 3, no. 1, pp. 134–141, Jan. 2012.

[2] J. Dowell, S. Weiss, D. Hill, and D. Infield, "Short-term spatio-temporal prediction of wind speed and direction," *Wind Energy*, vol. 17, no. 12, pp. 1945–1955, Dec. 2014.

[3] J. Z. Wang, Y. Wang, and P. Jiang, "The study and application of a novel hybrid forecasting model - A case study of wind speed forecasting in China," *Appl. Energy*, vol. 143, pp. 472–488, Apr. 2015.

[4] M. Lei, L. Shiyan, J. Chuanwen, L. Hongling, and Z. Yan, "A review on the forecasting of wind speed and generated power," *Renew. Sustain. Energy Rev.*, vol. 13, no. 4, pp. 915–920, May 2009.

[5] O. Ait Maatallah, A. Achuthan, K. Janoyan, and P. Marzocca, "Recursive wind speed forecasting based on Hammerstein Auto-Regressive model," *Appl. Energy*, vol. 145, pp. 191–197, May 2015.

[6] Q. Hu, S. Zhang, M. Yu, and Z. Xie, "Short-Term Wind Speed or Power Forecasting With Heteroscedastic Support Vector Regression," *IEEE Trans. Sustain. Energy*, vol. 7, no. 1, pp. 241–249, Jan. 2016.

[7] X. Zhu and M. G. Genton, "Short-Term Wind Speed Forecasting for Power System Operations," *Int. Stat. Rev.*, vol. 80, no. 1, pp. 2–23, Apr. 2012.

[8] G. Box and G. Jenkins, *Time series Analysis: Forecasting and Control*. San Francisco, California: Holden-Day, 1976.

[9] S. Haykin, *Neural Networks and Learning Machines*, 3rd ed. New Jersey: Pearson, 2009.

[10] J. M. Mendel, "Fuzzy logic systems for engineering: A tutorial," *Proc. IEEE*, vol. 83, no. 3, pp. 345–377, 1995.

[11] J. Jung and R. P. Broadwater, "Current status and future advances for wind speed and power forecasting," *Renew. Sustain. Energy Rev.*, vol. 31, pp. 762–777, Mar. 2014.

[12] T. Mitchell, *Machine Learning*. McGraw-Hill, 1997.

[13] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. Springer, 2009.

[14] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, Aug. 1997.

[15] L. Breiman, "Random Forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[16] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[17] A. Sfetsos, "A comparison of various forecasting techniques applied to mean hourly wind speed time series," *Renew. Energy*, vol. 21, no. 1, pp. 23–35, Sep. 2000.

[18] A. Müller and S. Guido, *Introduction to Machine Learning with Python*. Sebastopol, California: O'Riley Media, 2017.

APÊNDICE

A seguir, são apresentados os códigos com implementação das 21 DTs do trabalho (Figura 7), código com implementação de uma RF simples (Figura 8) e de um simples Adaboost de DTs (Figura 9), criados com base nas em soluções presentes em [18].

https://proceedings.science/p/85339?lang=pt-br

```
# Implement Decision Trees
treeN_train_score = []
treeN_val_score = []
treeN_test_score = []
treeN_train_rmse = []
treeN_val_rmse = []
treeN_test_rmse = []
treeN_feature_importances = []
settings = range(3, 23)
treeN_settings = np.array(settings, dtype=float)

for max_depth in treeN_settings:
    # build the model
    treeN = DecisionTreeRegressor(max_depth=max_depth)
    treeN = treeN.fit(X_train, y_train)

    # store metrics
    treeN_train_score.append(treeN.score(X_train, y_train))
    treeN_val_score.append(treeN.score(X_val, y_val))
    treeN_train_rmse.append(math.sqrt(mean_squared_error(y_train, treeN.predict(X_train))))
    treeN_val_rmse.append(math.sqrt(mean_squared_error(y_val, treeN.predict(X_val))))

    # Store feature importances
    treeN_feature_importances.append(treeN.feature_importances_)
```

Figura 7 – Principal código para implementação das DTs com scikit-learn

---

```
from sklearn.ensemble import RandomForestRegressor

# Implement Random Forest
forest = RandomForestRegressor(random_state=0, max_depth=20, n_estimators=10)

# fit model
forest = forest.fit(X_train, y_train)

# predict
forest_train_predict = forest.predict(X_train)
forest_val_predict = forest.predict(X_val)

# store metrics
forest_train_score = forest.score(X_train, y_train)
forest_val_score = forest.score(X_val, y_val)
forest_train_rmse = math.sqrt(mean_squared_error(y_train, forest_train_predict))
forest_val_rmse = math.sqrt(mean_squared_error(y_val, forest_val_predict))
```

Figura 8 – Principal código para implementação de uma única RF com scikit-learn

---

```
from sklearn.ensemble import AdaBoostRegressor

# Implement Random Forest
adatree = AdaBoostRegressor(DecisionTreeRegressor(max_depth=5),
                            random_state=2, n_estimators=20, learning_rate=1)

# fit model
adatree = adatree.fit(X_train, y_train)

# predict
adatree_train_predict = adatree.predict(X_train)
adatree_val_predict = adatree.predict(X_val)

# store metrics
adatree_train_score = adatree.score(X_train, y_train)
adatree_val_score = adatree.score(X_val, y_val)
adatree_train_rmse = math.sqrt(mean_squared_error(y_train, adatree_train_predict))
adatree_val_rmse = math.sqrt(mean_squared_error(y_val, adatree_val_predict))
```

Figura 9 – Principal código para implementação de uma única AdaBoost de DTs com scikit-learn

https://proceedings.science/p/85339?lang=pt-br