# Árvore de Decisão e Floresta Aleatória

Israel de Oliveira

11° Encontro: Machine Learning
Grupo Meetup Machine Learning Porto Alegre
Parque Científico e Tecnológico da PUCRS (Tecnopuc)

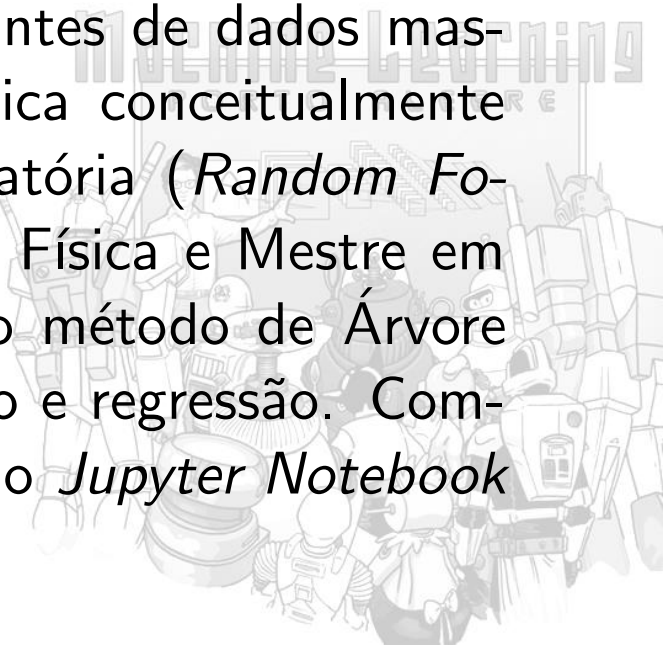
20 de novembro, 2018

# Sumário



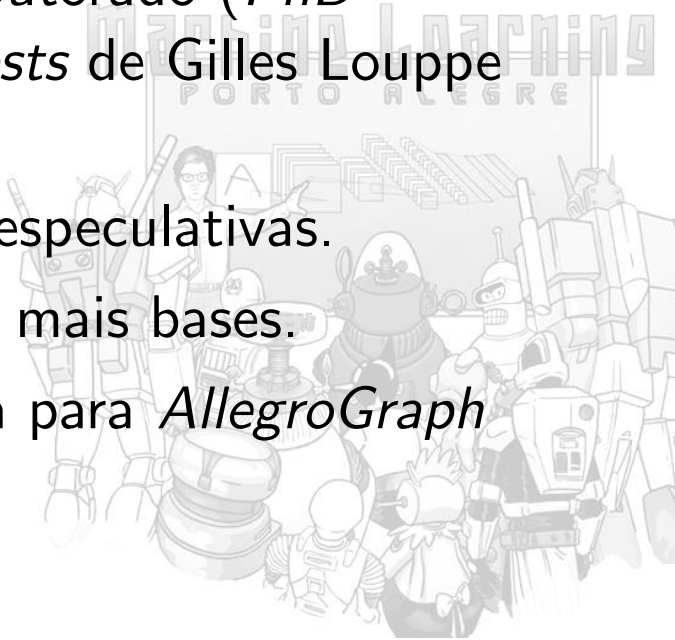
# Resumo

Com o aumento da popularidade da ciência de dados, técnicas de reconhecimentos de padrões, baseadas em aprendizado de máquina, servem como extratoras de informações relevantes de dados massivos. Dentre as diversas técnicas, uma técnica conceitualmente simples mas muito poderosa é a Floresta Aleatória (*Random Forest*). Técnico em Informática, Licenciado em Física e Mestre em Eng. Elétrica, Israel de Oliveira apresentará o método de Árvore Decisória e Floresta Aleatória para classificação e regressão. Compartilhará algumas noções e melhorias utilizando *Jupyter Notebook* e *Python*.



# Sobre o projeto

- O desenvolvimento e implementação dos métodos de Árvore Decisória (*Decision Tree*) e Floresta Aleatória (*Random Forest*) são baseados na dissertação de doutorado (*PhD dissertation*) *Understanding Random Forests* de Gilles Louppe [1] [2]
- É um laboratório: aplicação de variações especulativas.
- Resultados interessantes, cabe testar com mais bases.
- Objetivo final: biblioteca floresta aleatória para *AllegroGraph* [3].

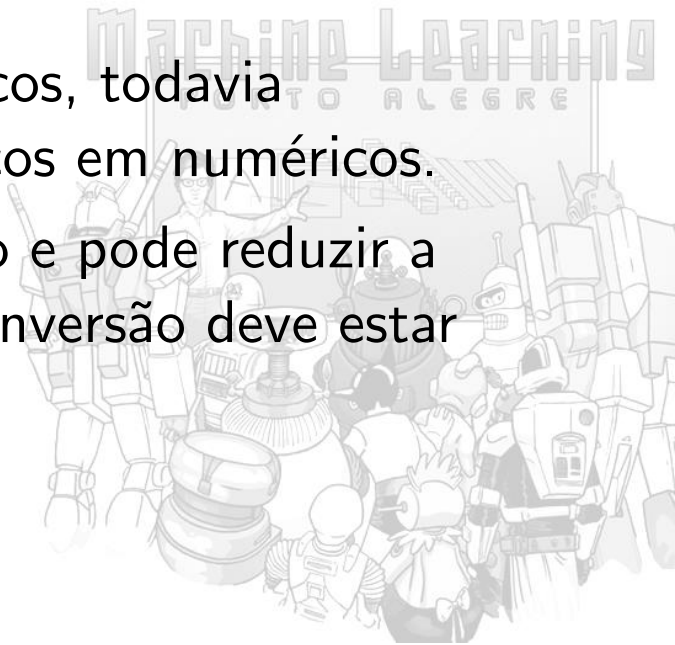


# Objetivos da apresentação

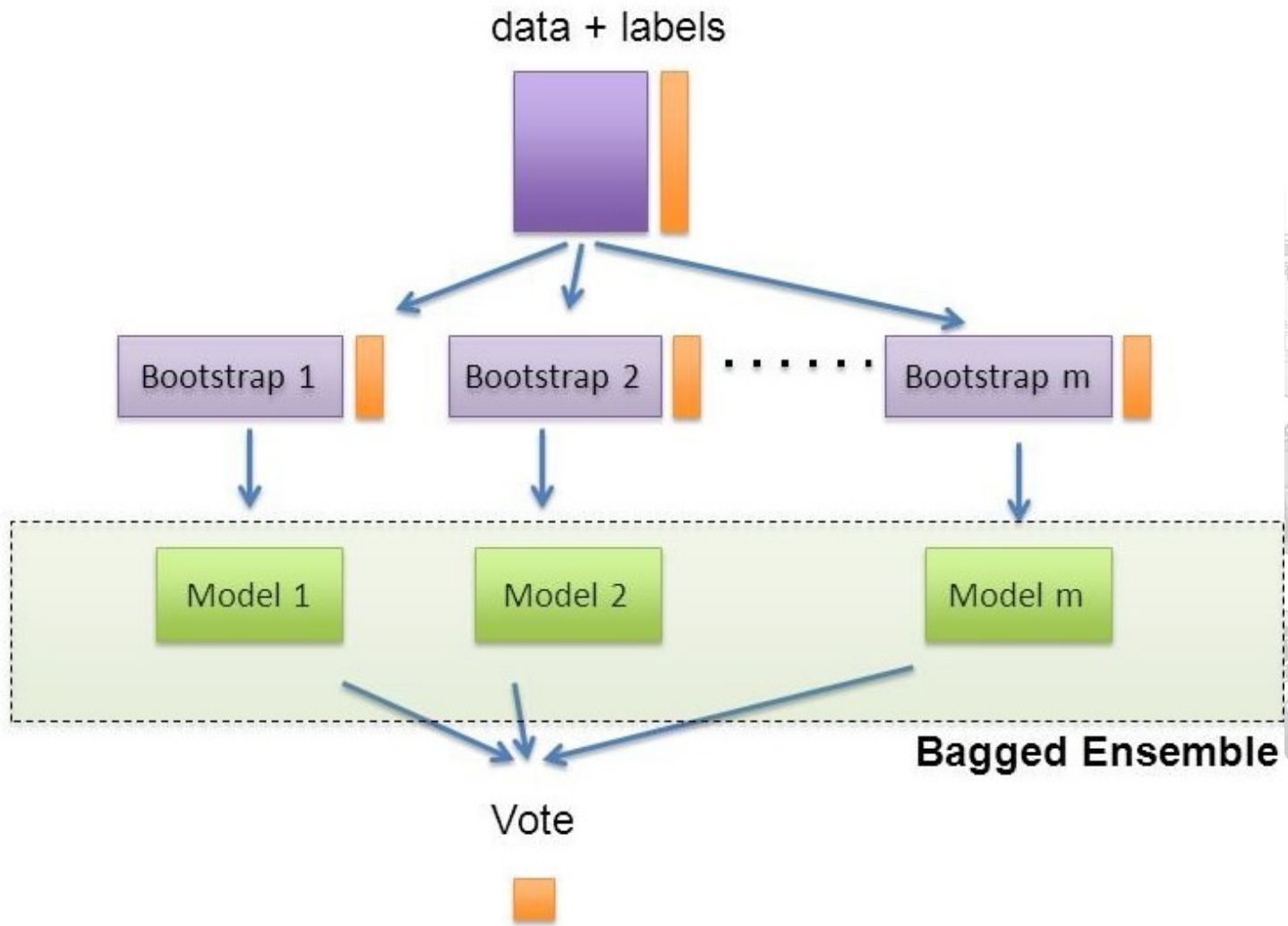
- Árvores de Decisão. Classificação (no projeto tem regressão).
- Método Ensemble (grupo, conjunto) e *Bagging* ("ensacamento- **B***ootstrap* **agg***regat***ing**).
- Floresta Aleatória.

# Árvore de Decisão

- Classificação: tipo de dados (*features*) categóricos e numéricos. Dados numéricos podem ser categorizados (por métodos de quantização, p.e.).
- Regressão: também categóricos e numéricos, todavia recomenda-se converter os tipos categóricos em numéricos.
- Usar mesmo tipo facilita a implementação e pode reduzir a complexidade, entretanto o método de conversão deve estar sempre junto ao modelo.
- Vá para o *Jupyter Notebook*!



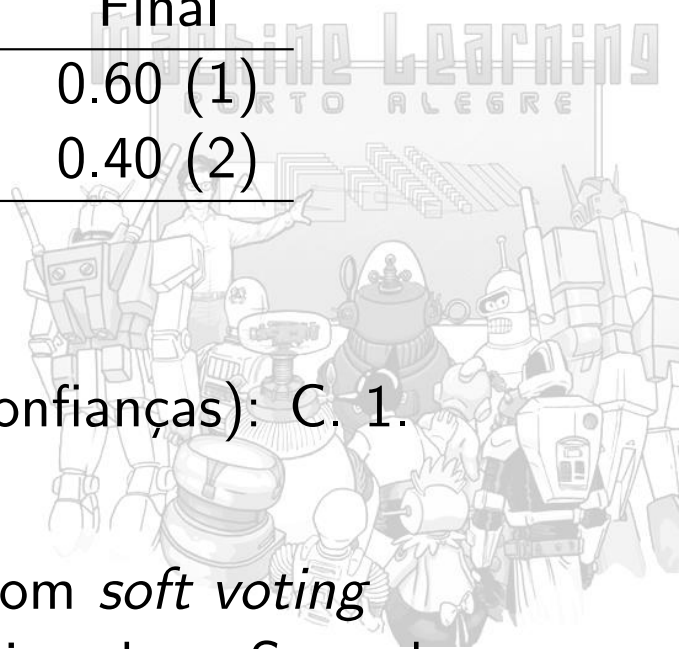
# Bagging [4]


data + labels
Bootstrap 1
Bootstrap 2
Bootstrap m
Model 1
Model 2
Model m
Bagged Ensemble
Vote

# Ensemble

Tabela 1: *Soft (hard) voting*

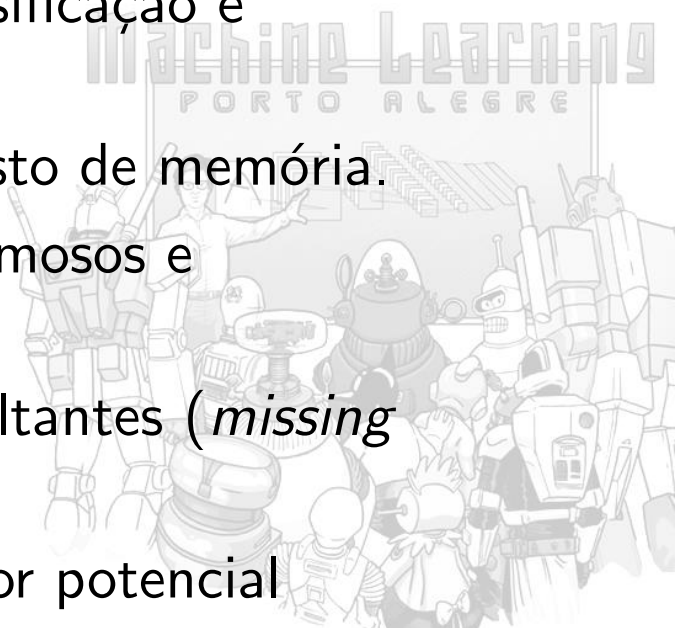
| | M. 1 | M. 2 | M. 3 | Final |
|---|---|---|---|---|
| C. 1 | 0.40 (0) | 0.45 (0) | 0.95 (1) | 0.60 (1) |
| C. 2 | 0.60 (1) | 0.55 (1) | 0.05 (0) | 0.40 (2) |




- *Soft voting* (combina as probabilidades/confianças): C. 1.
- *Majority (hard) voting*: C. 2
- Em geral, se obtém melhores resultados com *soft voting* usando árvores decisórias. Um caso negativo: base Szeged Weather [5]

# Floresta Aleatória (*Random Forest*) [1]

- Gerar árvores com discriminantes variados, em conjunto oferecem melhor generalização.
- Melhor relação entre performance de classificação e complexidade.
- Possibilidade de paralelização e menor custo de memória.
- Maior viabilidade em aplicar à dados volumosos e diversificados.
- Pode ser usado para predição de dados faltantes (*missing data*).
- Avaliar os dados: quais features com maior potencial discriminante. Servindo para comparar com outros métodos estatísticos. Análise de variância, PCA etc.
- Volte para o Jupyter Notbook!

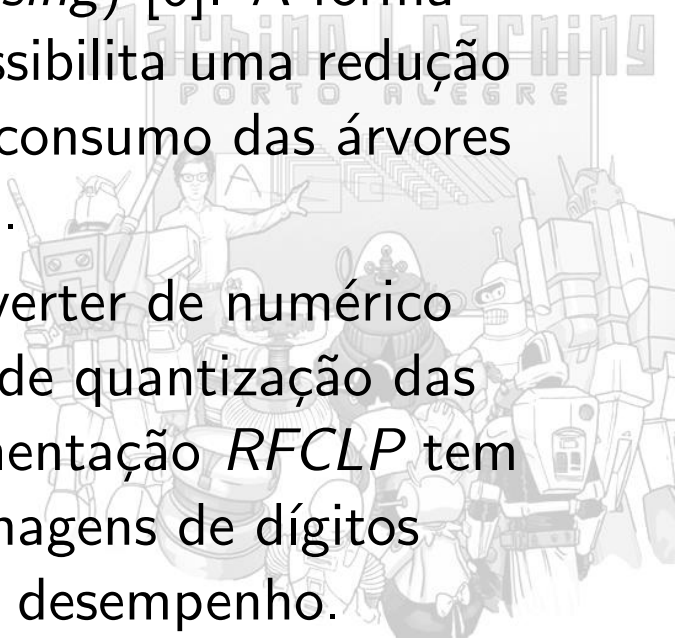


# Extras

- Poda (*prune*): descarte das árvores com base em critérios, p.e.: precisão, tamanho etc. Pode-se utilizar o mesmo gerador de lotes (*batchs*) da base de treino ou segmentar previamente a base de treino em duas, uma para gerar as árvores e outra para podar.
- Poda dinâmica: inicia com um critério "alto" (mais restrito), dado um limite de podas, reduz gradualmente o critério. Automatiza a busca pelo critério ótimo de poda.
- Com a poda, a RFS e com um limite no total de *features* a serem escolhidas, causa uma "seleção natural" de *features*. Em suma, pode-se obter alta precisão ao custo de se precisar de mais tempo.
- Descarte de *features* "ruins" reduz complexidade.

# Extras

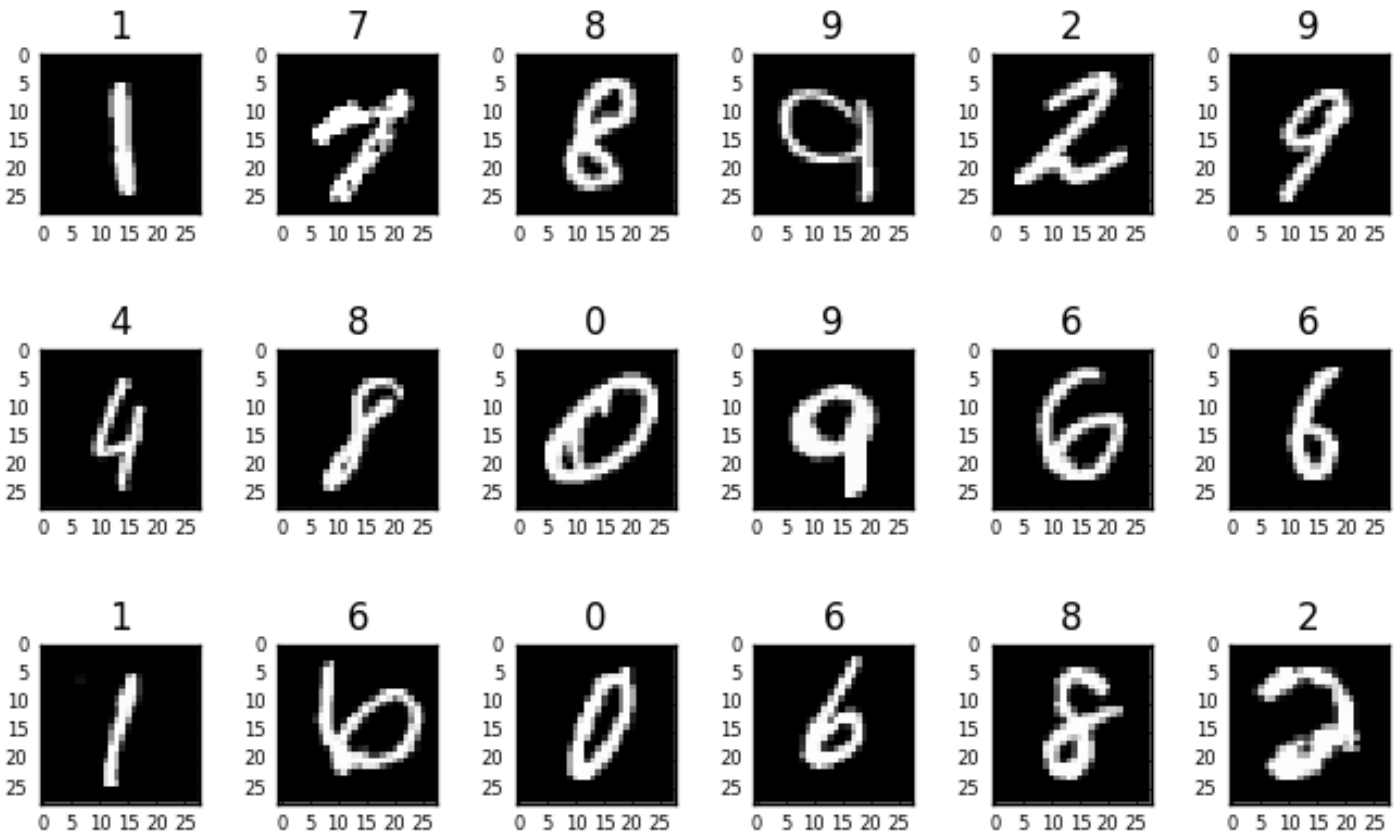
- Como árvores decisórias são listas compostas de listas, é possível obter melhor performance (menor tempo de processamento) usando LISP (*List Processing*) [6]. A forma de se trabalhar com as listas em LISP possibilita uma redução da quantidade de operações na criação e consumo das árvores (passo de treino e teste, respectivamente).
- Quantização no espaço das *features*: converter de numérico para categórico. Semelhante ao processo de quantização das imagens nos espaço das cores. Na implementação *RFCLP* tem uma quantização dos tons de cinza das imagens de dígitos (MNIST) com incremento significativo de desempenho. Critério de quantização: duas medianas sucessivas (4 categorias).



# Extras

- Vantagens: redução da complexidade, pois não há a necessidade de avaliar *features* numéricas repetidas vezes e aliado a poda e a RFS, utiliza-se uma pequena parte das *features*. Com imagens, no caso da MNIST, utiliza-se em torno de 150 pixels dos 784 (aprox. 20%).
- Desvantagens: necessita de um processamento prévio com uma parte representativa da base, ou seja, a quantização pode demandar um tempo maior. No caso da MNIST, usando 10% da base, dobra o tempo total de treino. É atrelado ao modelo a tabela de quantização, cada amostra, para cada *feature* de cada árvore, deve ser quantizada.
- Na prática, se tem ganho significativo de desempenho, pois os passo de poda possui menor complexidade (se constrói mais árvores no mesmo tempo).

# MNIST [7]



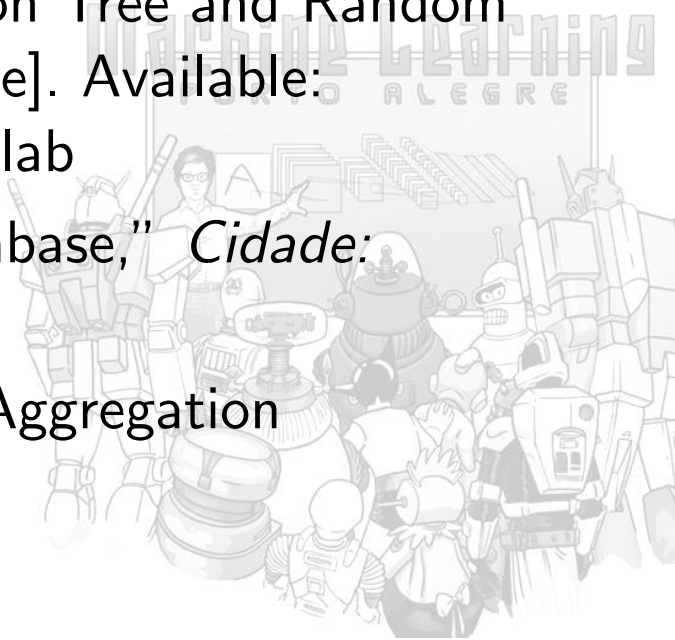

# Referências I

[1] G. Louppe, "Understanding random forests: From theory to practice," 10 2014, arXiv:1407.7502.

[2] I. de Oliveira. (2018) Laboratory of Decision Tree and Random Forest (GitHub: random-forest-lab). [Online]. Available: https://github.com/ysraell/random-forest-lab

[3] J. Aasman, "Allegro graph: Rdf triple database," *Cidade: Oakland Franz Incorporated*, vol. 17, 2006.

[4] DynamicWebPaige. (2018) Bootstrapped Aggregation (Bagging). [Online]. Available: https://twitter.com/DynamicWebPaige

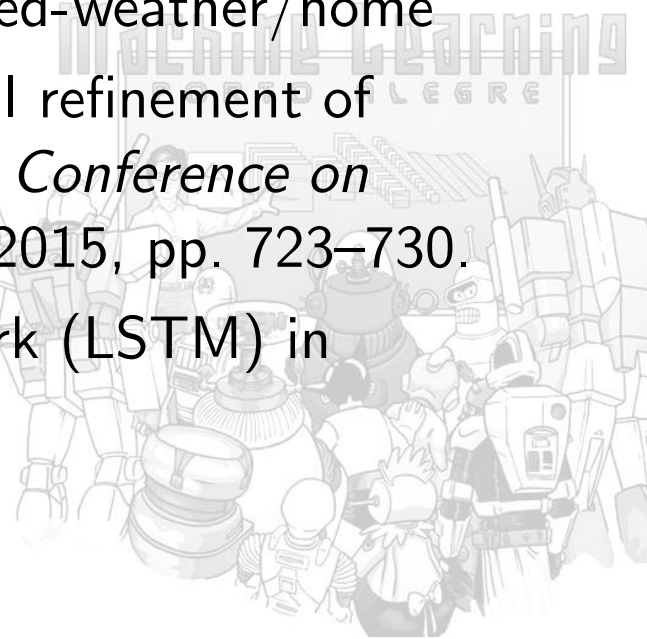


# Referências II

[5] Kaggle Inc. (2018) Weather in Szeged 2006-2016: Hourly/daily summary with temperature, pressure, wind speed and more. [Online]. Available: https://www.kaggle.com/budincsevity/szeged-weather/home

[6] S. Ren, X. Cao, Y. Wei, and J. Sun, "Global refinement of random forest," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 723–730.

[7] S. Zafrany. (2018) Reccurent Neural Network (LSTM) in Tensorflow. [Online]. Available: https://samyzaf.com/ML/rnn1/rnn1.html



**Obrigado pela atenção.**

- E-mail: prof.israel@gmail.com
- Linkedin: www.linkedin.com/in/prof-igo/
- GitHub: ysraell

(a) e-mail e GitHub

(b) Linkedin