Exploring Tools for Interpretable Machine Learning

Dr. Juan Orduz

PyData Global 2021

PyData Global

Outline

Introduction

Aim and Scope of the Talk

We want to test explore various techniques to get a better understanding on how machine learning (ML) models generate predictions and how features interact with each other.

Important!

- Domain knowledge on the problem.
- Understanding on the input data.
- Understanding the logic behind the ML algorithms.

Introduction

Aim and Scope of the Talk

We want to test explore various techniques to get a better understanding on how machine learning (ML) models generate predictions and how features interact with each other.

Important!

- Domain knowledge on the problem.
- Understanding on the input data.
- Understanding the logic behind the ML algorithms.

How? We are going to work out a concrete example.

References

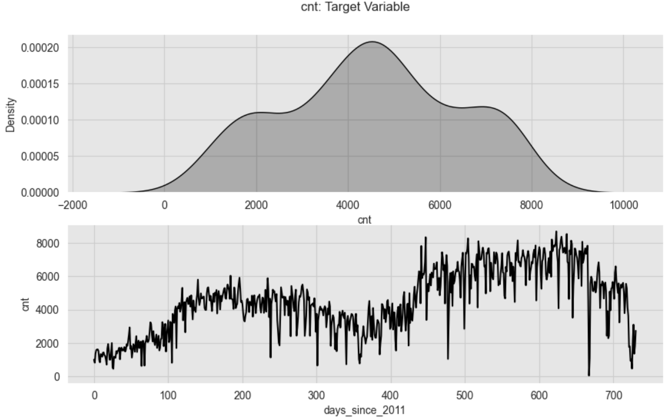
This talk is based on my blog post ([9]), which itself is based on these two amazing references:

- Interpretable Machine Learning, A Guide for Making Black Box Models Explainable by Christoph Molnar ([7])
- Interpretable Machine Learning with Python by Serg Masís ([5])

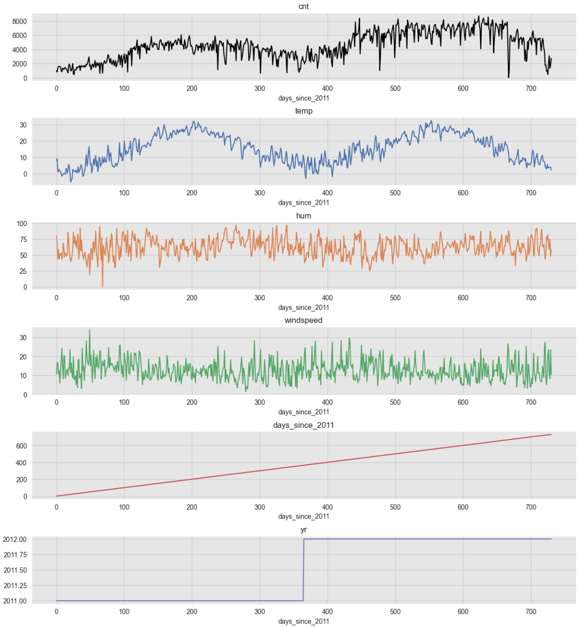
Remark: Interpretable ML $\neq$ Causality (see [2], [3], [6] and [8])

Target Variable - cnt: Daily Bike Rents

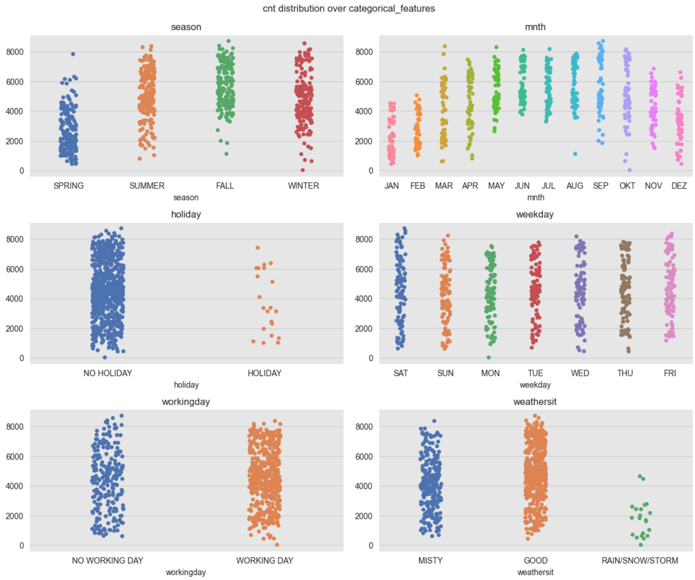
	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	hum	windspeed	cnt	days_since_2011
0	SPRING	2011	JAN	NO HOLIDAY	SAT	NO WORKING DAY	MISTY	8.175849	80.5833	10.749882	985	0
1	SPRING	2011	JAN	NO HOLIDAY	SUN	NO WORKING DAY	MISTY	9.083466	69.6087	16.652113	801	1
2	SPRING	2011	JAN	NO HOLIDAY	MON	WORKING DAY	GOOD	1.229108	43.7273	16.636703	1349	2
3	SPRING	2011	JAN	NO HOLIDAY	TUE	WORKING DAY	GOOD	1.400000	59.0435	10.739832	1562	3
4	SPRING	2011	JAN	NO HOLIDAY	WED	WORKING DAY	GOOD	2.666979	43.6957	12.522300	1600	4



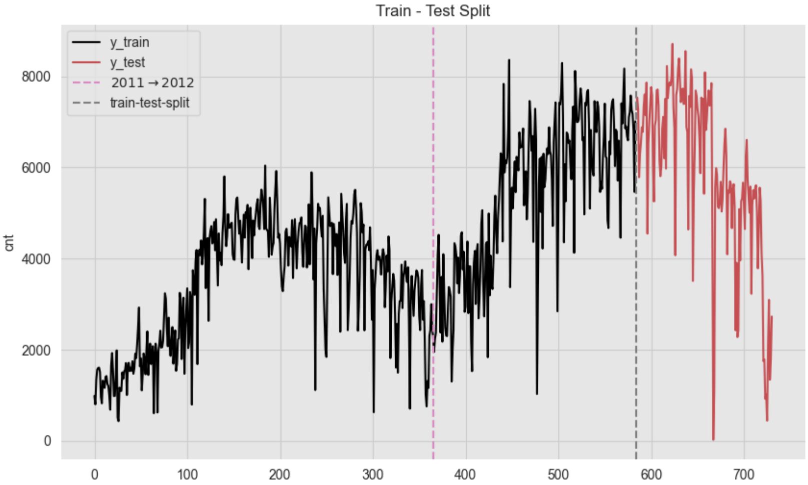
Continuous Regressors



Categorical Regressors



Train-Test Split



Models

Two model flavours

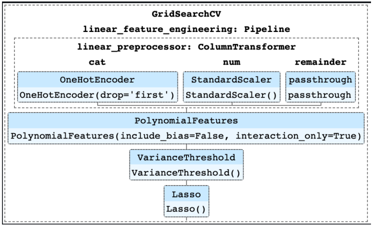


Figure 1: **Linear model** Lasso + second order polynomial interactions ([10]).

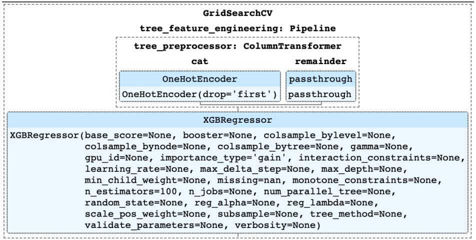
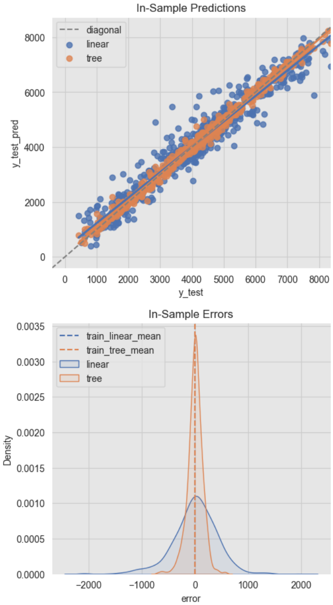
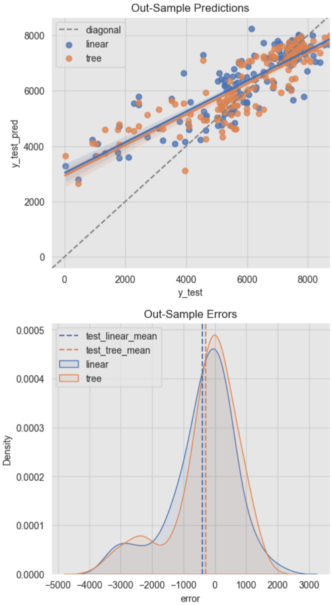


Figure 2: **Tree based model** XGBoost regression model ([1]).

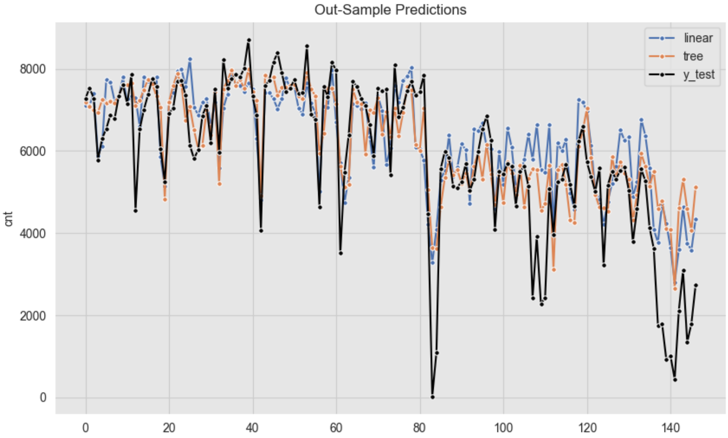
PyData Global

Out of sample performance - Errors Distribution





Out of sample performance - Predictions



β coefficients

See [7, Section 5.1]

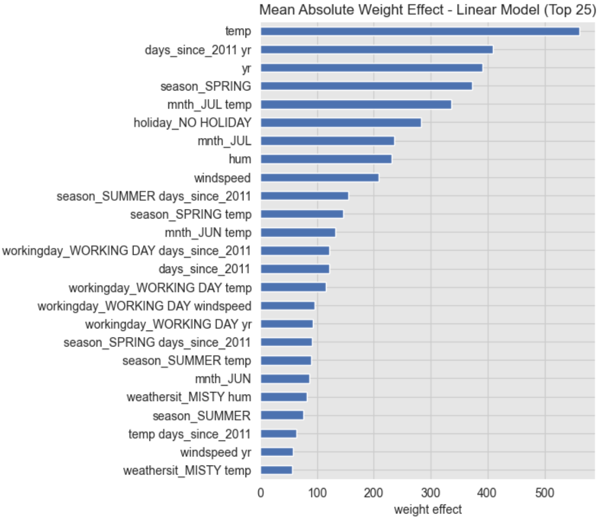
$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p + \varepsilon, \quad \text{where} \quad \varepsilon \sim N(0, \sigma^2)$$

	linear_features	coef_	abs_coef_
0	mnth_JUL temp	-2305.096894	2305.096894
1	mnth_JUL	2227.672335	2227.672335
2	weathersit_RAIN/SNOW/STORM	-1710.469071	1710.469071
3	mnth_JUN temp	-1299.644413	1299.644413
4	season_SPRING	-1279.629779	1279.629779
5	mnth_JUN	845.229031	845.229031
6	temp	646.609622	646.609622
7	mnth_AUG temp	-523.011653	523.011653
8	season_SUMMER temp	489.319256	489.319256
9	weathersit_RAIN/SNOW/STORM temp	-482.660271	482.660271
10	season_SPRING temp	465.512410	465.512410
11	days_since_2011 yr	465.079169	465.079169
12	weekday_SUN weathersit_RAIN/SNOW/STORM	-462.286059	462.286059
13	season_SUMMER days_since_2011	454.137278	454.137278
14	mnth_MAY temp	-445.268148	445.268148
15	season_SUMMER weathersit_MISTY	-408.809531	408.809531
16	mnth_MAY weathersit_MISTY	404.790954	404.790954
17	yr	403.199142	403.199142
18	season_SUMMER weathersit_RAIN/SNOW/STORM	-394.157306	394.157306
19	mnth_DEZ temp	363.222114	363.222114

Weight Effects $\beta_i x_i$

Figure 3: For each data instance i and each feature x_k we compute the product $\beta_k x_k^{(i)}$ to get the weight effect.

Weight Effects Importance $w_k = \frac{1}{n}\sum_{i=1}^{n} |\beta_k x_k^{(i)}|$



PyData Global

Weight Effects: Temperature (z-transform)

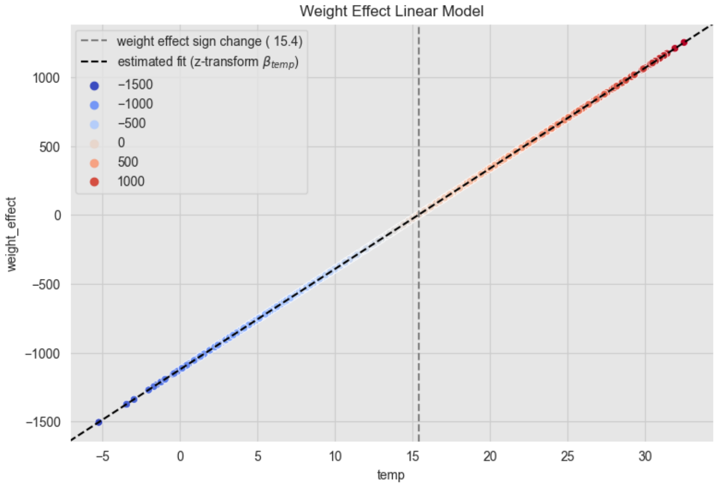


Figure 4: This plot just shows the effect of the linear term *temp* and not the interactions.

Weight Effects: Interactions

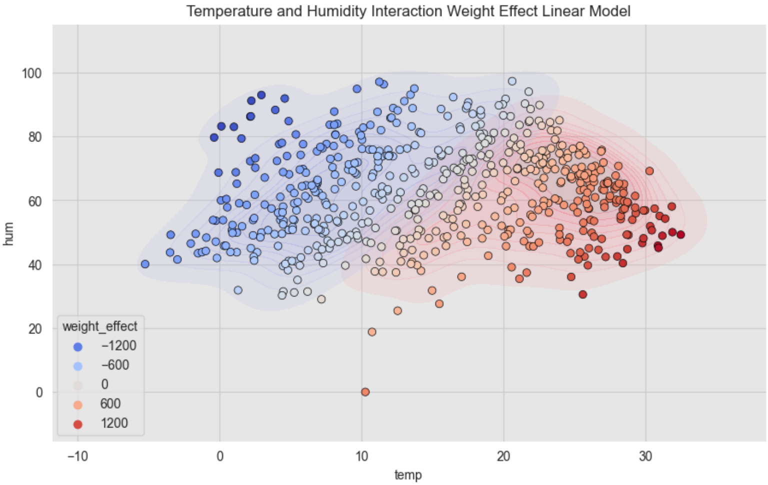


Figure 5: We can visualize the interaction between *temp* and *hum* by computing the total weight effect $\beta_{temp}x_{temp} + \beta_{hum}x_{hum} + \beta_{temp \times hum}x_{temp}x_{hum}$.

Explaining Individual Predictions

Let us see weight effects of the linear model for data observation 284

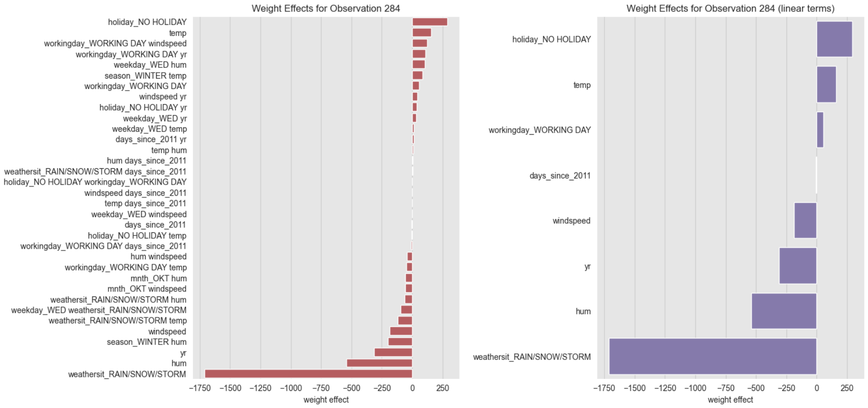
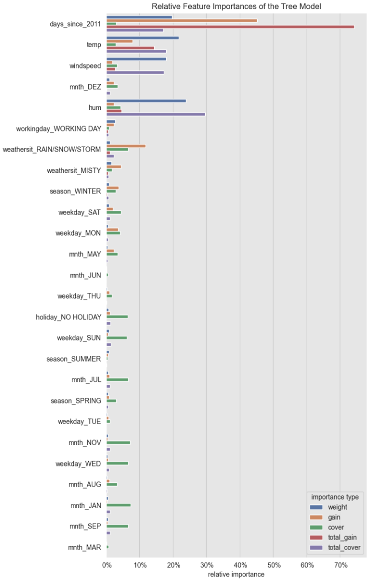


Figure 6: Left: All weight effects. Right: Weight effects of the linear terms.

Feature Importance Metrics: XGBoost ([1])

- **Gain:** improvement in accuracy brought by a feature to the branches it is on.
- **Cover:** measures the relative quantity of observations concerned by a feature.
- **Frequency / Weight:** just counts the number of times a feature is used in all generated trees.



Partial Dependence Plot (PDP) & Individual Conditional Expectation (ICE) ([7, Section 8.1 & 9.1])

- The partial dependence plot shows the marginal effect one or two features have on the predicted outcome of a machine learning model.
- For example, given a trained model $\hat{f}$, we compute for $temp = 8$

$$\hat{f}_{temp}(temp = 8) = \frac{1}{146}\Big(\hat{f}(temp = 8, hum = 80, \cdots) + \hat{f}(temp = 8, hum = 70, \cdots) + \cdots\Big)$$

Partial Dependence Plot (PDP) & Individual Conditional Expectation (ICE) ([7, Section 8.1 & 9.1])

- The partial dependence plot shows the marginal effect one or two features have on the predicted outcome of a machine learning model.
- For example, given a trained model $\hat{f}$, we compute for $temp = 8$

$$\hat{f}_{temp}(temp = 8) = \frac{1}{146}\Big(\hat{f}(temp = 8, hum = 80, \cdots) + \hat{f}(temp = 8, hum = 70, \cdots) + \cdots\Big)$$

- Individual conditional expectation (ICE) plot shows one line per instance.
- A PDP is the average of the lines of an ICE plot

PDP & ICE Examples (1D)

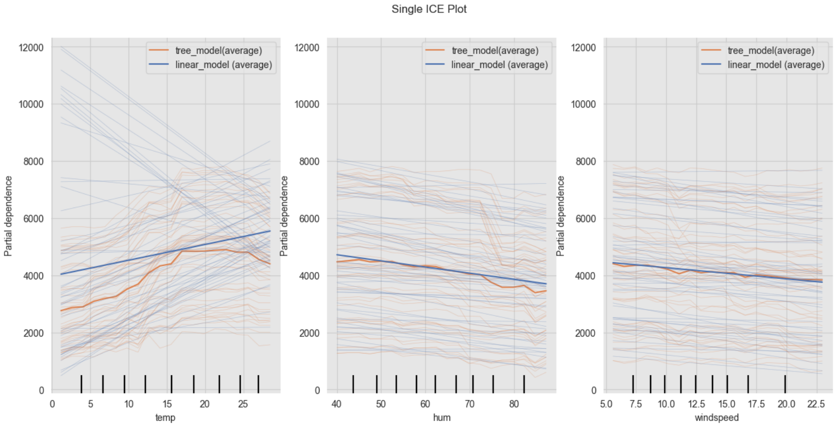
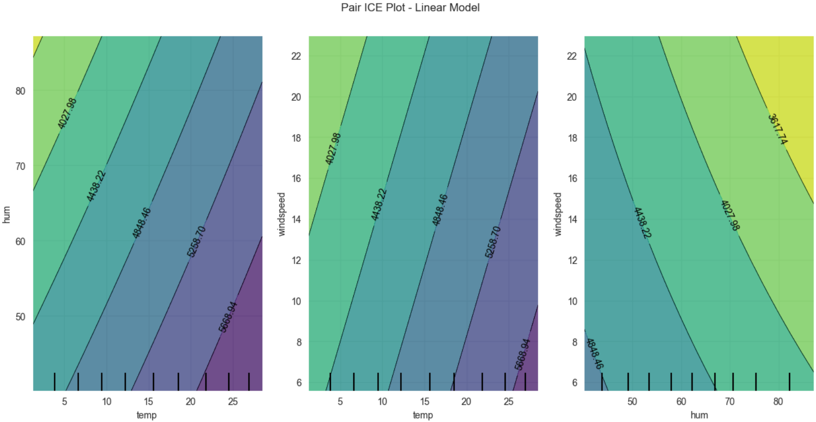


Figure 7: PDP & ICE plots for some numerical variables for the linear and XGBoost models.

PDP & ICE Examples (2D)



	linear_features	coef_	abs_coef_
6	temp	646.609622	646.609622
29	hum	-281.701209	281.701209
32	windspeed	-264.190697	264.190697
114	hum windspeed	-31.993914	31.993914
130	temp hum	15.127133	15.127133
155	temp windspeed	-0.000000	0.000000

Permutation Importance

See [7, Section 5.1]

Measures the increase in the prediction error of the model after we permuted the feature's values, which breaks the relationship between the feature and the true outcome ([7, Section 8.5]).

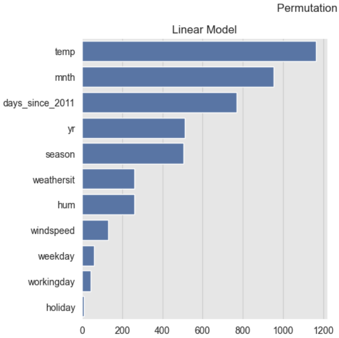


Figure 8: The permutation importance for these two models have *days_since_2011* and *temp* on their top 3 ranking, which partially explain the trend and seasonality components respectively (see [7, Figure 8.27]).

SHAP Values: Features as teams playing a game

Definition, see [4], and [5, Chapters 5 & 6] and [7, Section 9.6]

For each data instance x (e.g. temp=15, hum=60, windspeed=14)

- Sample coalitions $z' \in \{0, 1\}^M$, where M is the maximum coalition size.
  - Assume we select *temp* and *hum* from $\{temp, hum, windspeed\}$.

SHAP Values: Features as teams playing a game

Definition, see [4], and [5, Chapters 5 & 6] and [7, Section 9.6]

For each data instance x (e.g. temp=15, hum=60, windspeed=14)

- Sample coalitions $z' \in \{0, 1\}^M$, where M is the maximum coalition size.
  - Assume we select *temp* and *hum* from $\{temp, hum, windspeed\}$.
- Get prediction for each z'. For features not in the coalition we replace their values with random samples from the dataset.
  - E.g. for a data instance $temp = 15$ and $hum = 60$ we compute the prediction $\hat{f}(temp = 15, hum = 60, windspeed = 11) = 4000$.

SHAP Values: Features as teams playing a game

Definition, see [4], and [5, Chapters 5 & 6] and [7, Section 9.6]

For each data instance x (e.g. temp=15, hum=60, windspeed=14)

- Sample coalitions $z' \in \{0, 1\}^M$, where M is the maximum coalition size.
  - Assume we select *temp* and *hum* from $\{temp, hum, windspeed\}$.
- Get prediction for each z'. For features not in the coalition we replace their values with random samples from the dataset.
  - E.g. for a data instance $temp = 15$ and $hum = 60$ we compute the prediction $\hat{f}(temp = 15, hum = 60, windspeed = 11) = 4000$.
- Compute the weight for each z', with the SHAP kernel,

$$\pi_x(z') = \frac{(M-1)}{\binom{M}{|z'|}|z'|(M-|z'|)}$$

  - $M = 3, |z`| = 2 \Rightarrow \pi = (3-1)/(3 \times 2 \times (3-2)) = 1/3.$

PyData Global

SHAP Values: Features as teams playing a game

Definition, see [4], and [5, Chapters 5 & 6] and [7, Section 9.6]

For each data instance x (e.g. temp=15, hum=60, windspeed=14)

- Sample coalitions $z' \in \{0, 1\}^M$, where M is the maximum coalition size.
  - Assume we select *temp* and *hum* from $\{temp, hum, windspeed\}$.
- Get prediction for each z'. For features not in the coalition we replace their values with random samples from the dataset.
  - E.g. for a data instance $temp = 15$ and $hum = 60$ we compute the prediction $\hat{f}(temp = 15, hum = 60, windspeed = 11) = 4000$.
- Compute the weight for each z', with the SHAP kernel,

$$\pi_x(z') = \frac{(M-1)}{\binom{M}{|z'|}|z'|(M-|z'|)}$$

  - $M = 3, |z`| = 2 \Rightarrow \pi = (3-1)/(3 \times 2 \times (3-2)) = 1/3.$
- Fit weighted linear model and return Shapley values, i.e. the coefficients from the linear model. In this example $4000 = \phi_0 + \frac{1}{3}\phi_{temp} + \frac{1}{3}\phi_{hum} + \varepsilon$.

PyData Global

SHAP Values

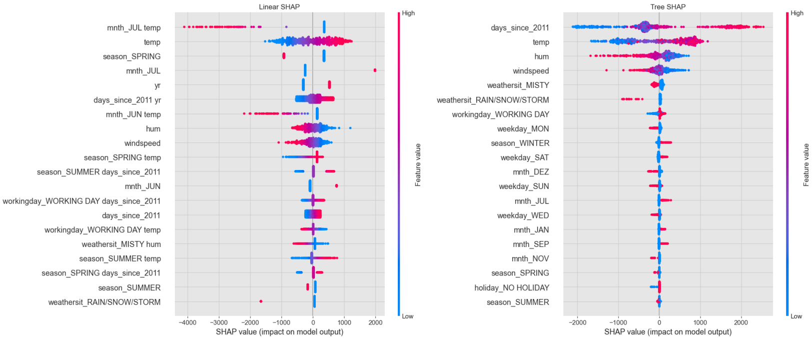
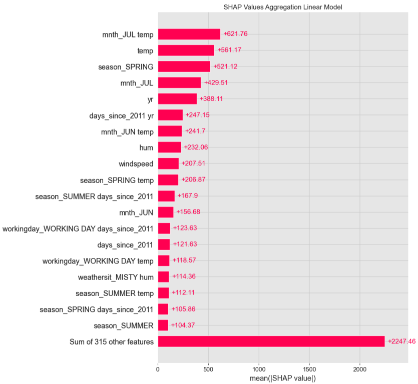
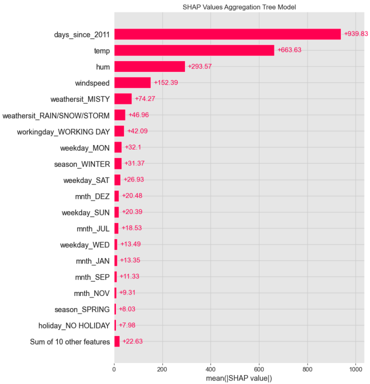


Figure 9: SHAP values per data instance. The x position of the dot is determined by the SHAP value of that feature, and dots "pile up" along each feature row to show density. Color is used to display the original value of a feature ([4]).

Mean Abs SHAP Values





SHAP Values: Temperature

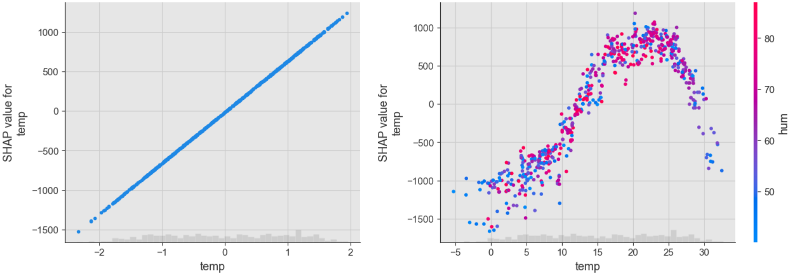


Figure 10: This figure shows the SHAP values as a function of temperature. Compare with Figure 7

SHAP Values: Observation 284

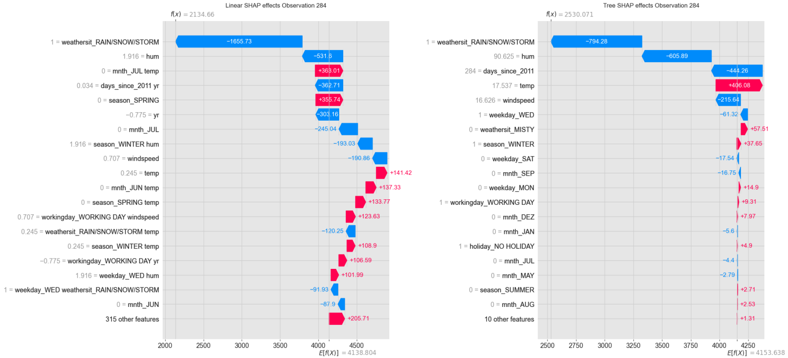


Figure 11: This *waterfall plot* shows how the SHAP values of each feature move the model output from our prior expectation under the background data distribution, to the final model prediction given the evidence of all the features ([4]). Compare with Figure 6.

References I

[1] Tianqi Chen and Carlos Guestrin.
XGBoost: A scalable tree boosting system.
In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

[2] Scott Cunningham.
Causal Inference: The Mixtape.
Yale University Press, 2021.

[3] Scott Lundberg.
Be careful when interpreting predictive models in search of causal insights.
`https://towardsdatascience.com/be-careful-when-interpreting-predictive-models-in-search-of-causa`
May 2021.

[4] Scott M Lundberg and Su-In Lee.
A unified approach to interpreting model predictions.
In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.

References II

[5] Serg Masís.
Interpretable Machine Learning with Python.
Packrat, 2021.
https://github.com/PacktPublishing/Interpretable-Machine-Learning-with-Python.

[6] Richard McElreath.
Statistical Rethinking: A Bayesian Course with Examples in R and Stan, 2nd Edition.
CRC Press, 2 edition, 2020.

[7] Christoph Molnar.
Interpretable Machine Learning.
2019.
https://christophm.github.io/interpretable-ml-book/.

[8] Christoph Molnar, Giuseppe Casalicchio, and Bernd Bischl.
Interpretable machine learning – a brief history, state-of-the-art and challenges, 2020.

[9] Juan Orduz.
Exploring tools for interpretable machine learning.
https://juanitorduz.github.io/interpretable_ml/, Jul 2021.

References III

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay.
Scikit-learn: Machine learning in Python.
Journal of Machine Learning Research, 12:2825–2830, 2011.

Thank You!

Contact

- https://juanitorduz.github.io
- github.com/juanitorduz
- juanitorduz
- juanitorduz@gmail.com