



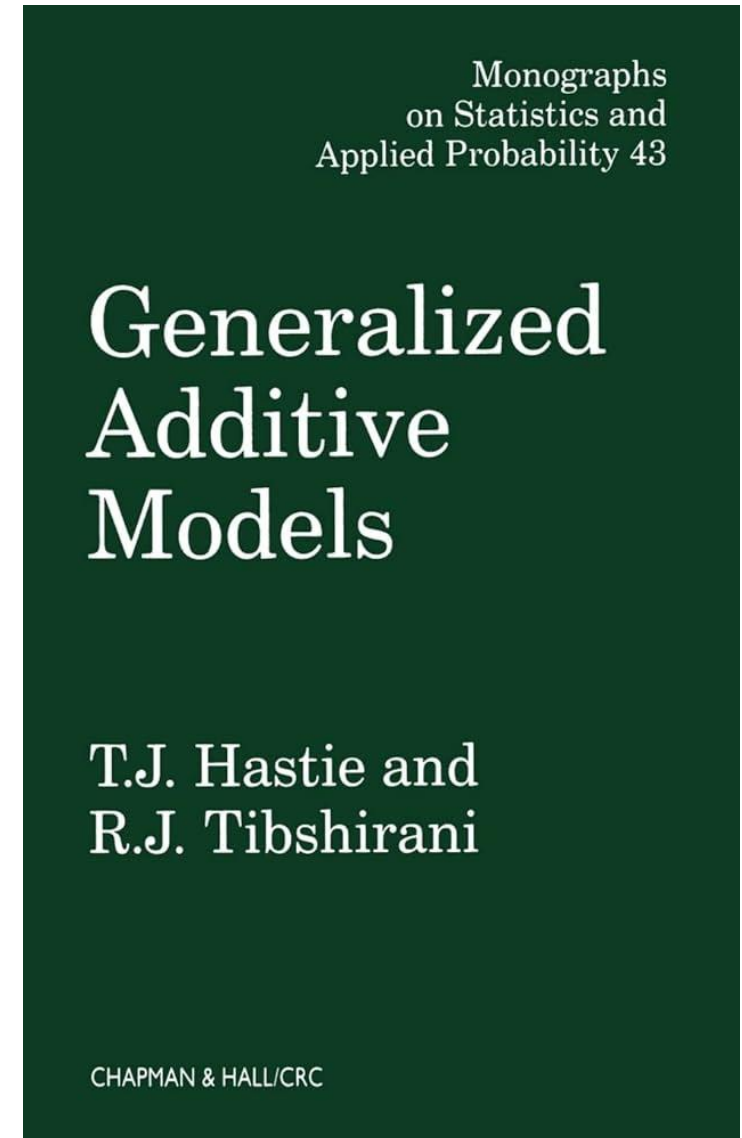
Pontifícia Universidade Católica do Rio de Janeiro
Centro Técnico Científico
Departamento de Engenharia Elétrica
ELE2345 - Data Science

Regressão: Generalized Additive Model (GAM)

Marcelo Vieira Aguiar

Contexto

- **Autores:** Trevor Hastie e Robert Tibshirani (1986)
 - Professores do departamento de estatística da Universidade de Stanford
- Generalized Additive Model – GAM (Modelo Aditivo Generalizado)
 - Algoritmo de Regressão
 - Aprende relações não lineares
 - Flexível
 - Generalização em relações complexas entre variáveis independentes e dependentes

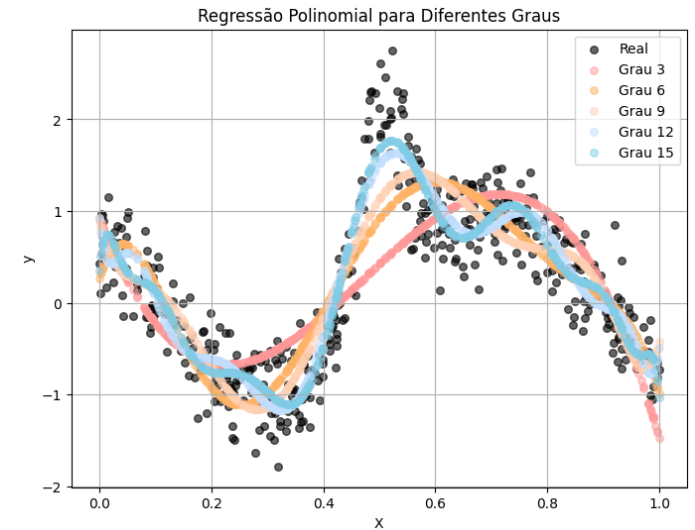
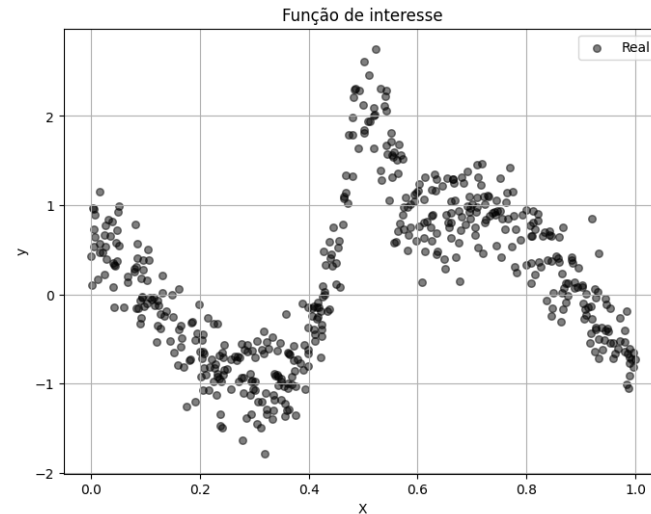


Contexto

- **Regressão Polinomial**

- Com aumento do grau, há overfitting.

$$\hat{y} = \beta_0 + \beta_1 x + \beta_2 x^2 + \dots + \beta_n x^n$$

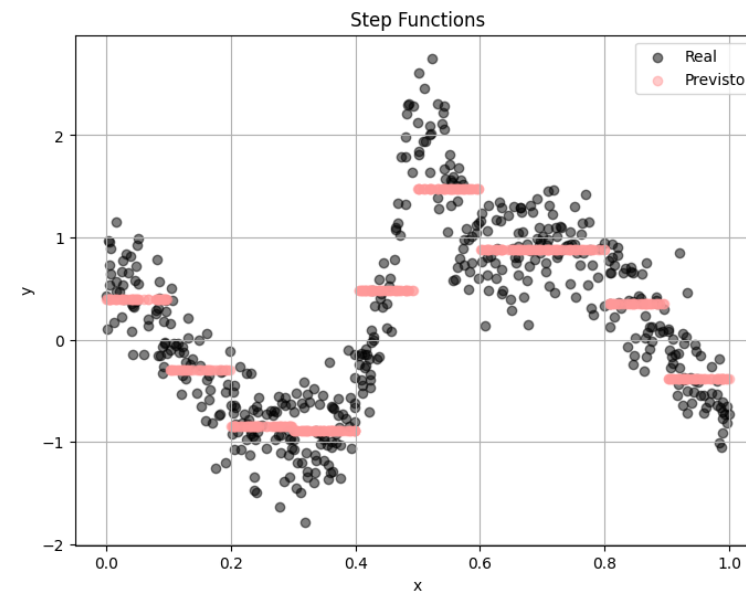


- **Regressão via Step Functions**

- Não se ajusta bem em dados não lineares

$$\hat{y} = \beta_0 + \beta_1 c_1(x) + \beta_2 c_2(x) + \dots + \beta_n c_n(x)$$

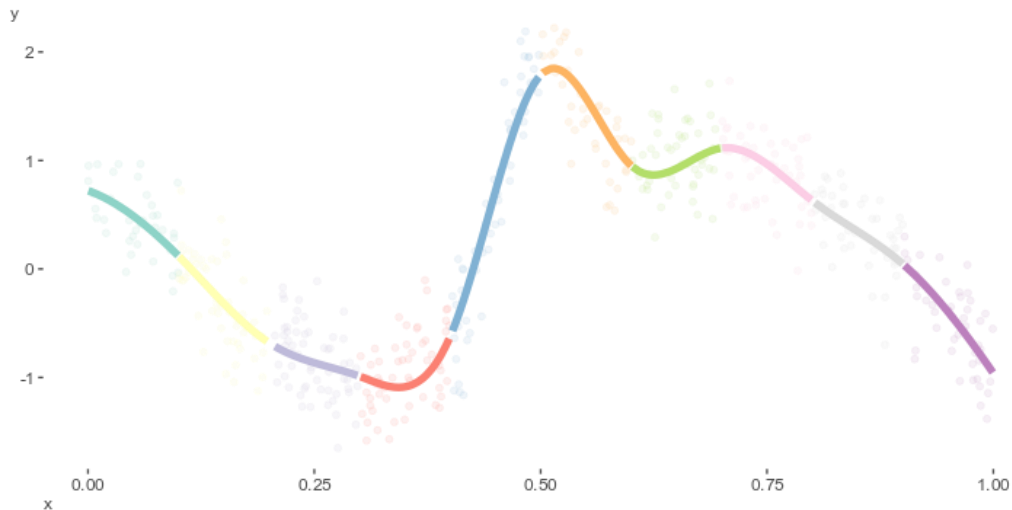
Se $(0.2 \leq x < 0.4, \text{ então } c_1(x) = 1 \Rightarrow \beta_1 = -0.9$



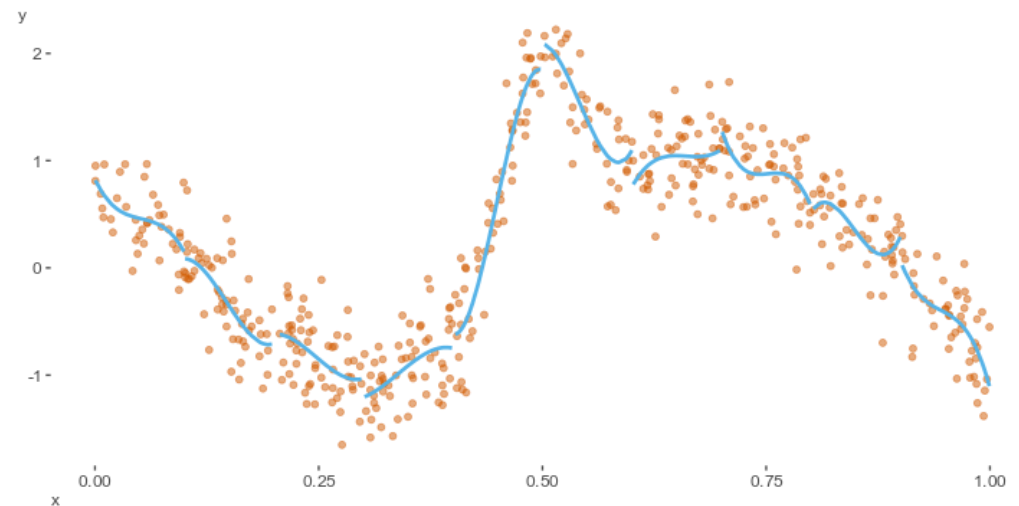
Contexto

- **Regressão por Splines**
 - Separa dados por regiões.
 - Cada região irá ser ajustada por um polinômio de terceiro grau.
 - Transição suave
 - Primeira e segunda derivadas contínuas no ponto de separação (restrição)
- Exemplo: 10 splines cubics

- **Com restrição**



- **Sem restrição**



GAM

- **Conceito**

- **Regressão linear com múltiplas variáveis**

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$



- **Generalized Additive Models**

$$\hat{y} = \beta_0 + \beta_1 f_1(x_1) + \beta_2 f_2(x_2) + \cdots + \beta_n f_n(x_n)$$

spline

Step function

Linear

- f_1, f_2 e f_n são as funções base e podem ser qualquer função (spline, step function, linear)
 - Se todas funções base forem linear, o GAM se torna uma regressão linear.
- Processo de ajuste dos β 's e das funções base é feito de forma conjunta pelo método dos mínimos quadrados
- Aplicações em diferentes áreas onde as variáveis são complexas e não lineares, como na modelagem de curvas e previsão de tendência.
 - Modelagem não linear mantendo a interpretabilidade do modelo.

Exemplo com código Python

- Biblioteca pyGAM
- Linear terms: `l()`

```
from pygam import GAM
from pygam import l
import matplotlib.pyplot as plt

modelo = GAM(l(0))

modelo.fit(X, y)

y_pred = modelo.predict(X)

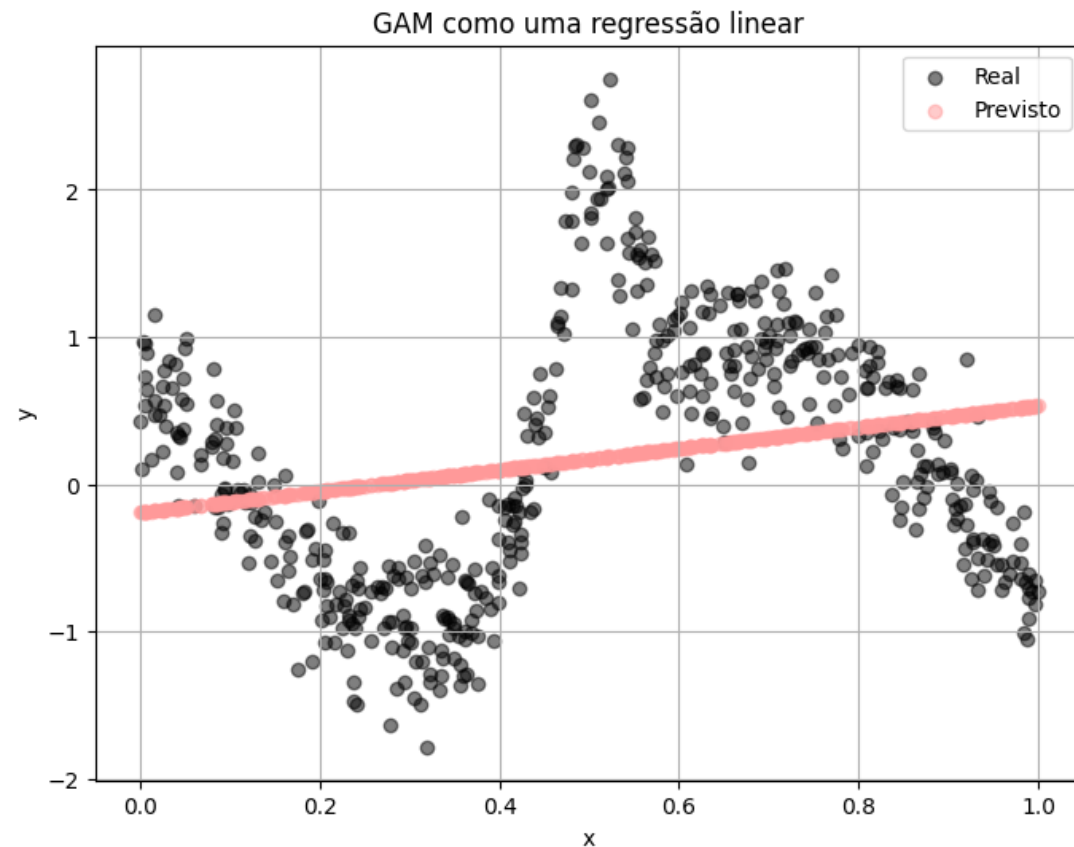
plt.figure(figsize=(8, 6))
plt.scatter(X, y, color='black', alpha=0.5)
plt.title("GAM como uma regressão linear")
plt.xlabel("y")
plt.ylabel("x")
plt.scatter(d["x"].values, y_pred, color='#FF9999', alpha=0.5)
plt.grid(True)
plt.show()
```

`modelo.coef_`

✓ 0.0s

`array([0.72586626, -0.19126074])`

[Código no GitHub](#)



Exemplo com código Python

- Factor terms: $f()$

```
modelo = GAM(f(0))

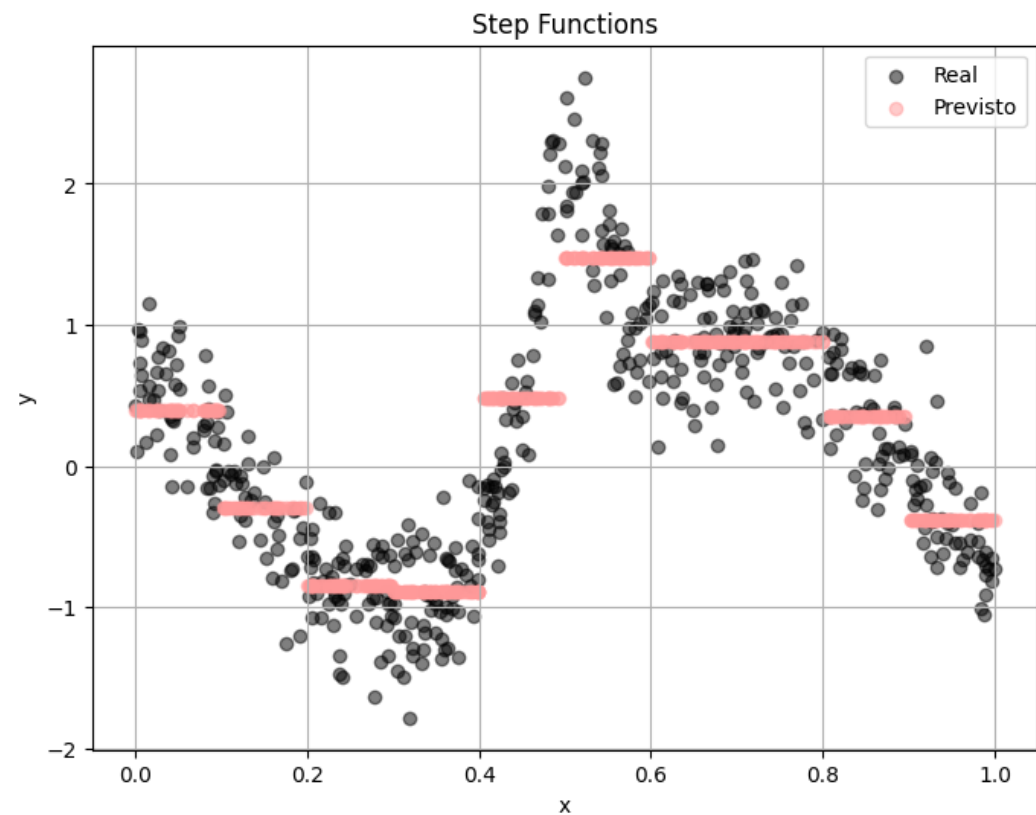
intervalos = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]
rotulos = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

# Categorizando x
X_rotulo = pd.cut(X, bins=intervalos, labels=rotulos, right=False)

modelo.fit(X_rotulo, y)

y_pred = modelo.predict(X_rotulo)

plt.figure(figsize=(8, 6))
plt.scatter(X, y, color='black', alpha=0.5)
plt.scatter(X, y_pred, color='#FF9999', alpha=0.5)
plt.title("Step Functions")
plt.xlabel("x")
plt.ylabel("y")
plt.grid(True)
plt.show()
```



Exemplo com código Python

- Spline terms: `s()`

```
modelo = GAM(s(0, n_splines=16))

modelo.fit(X, y)

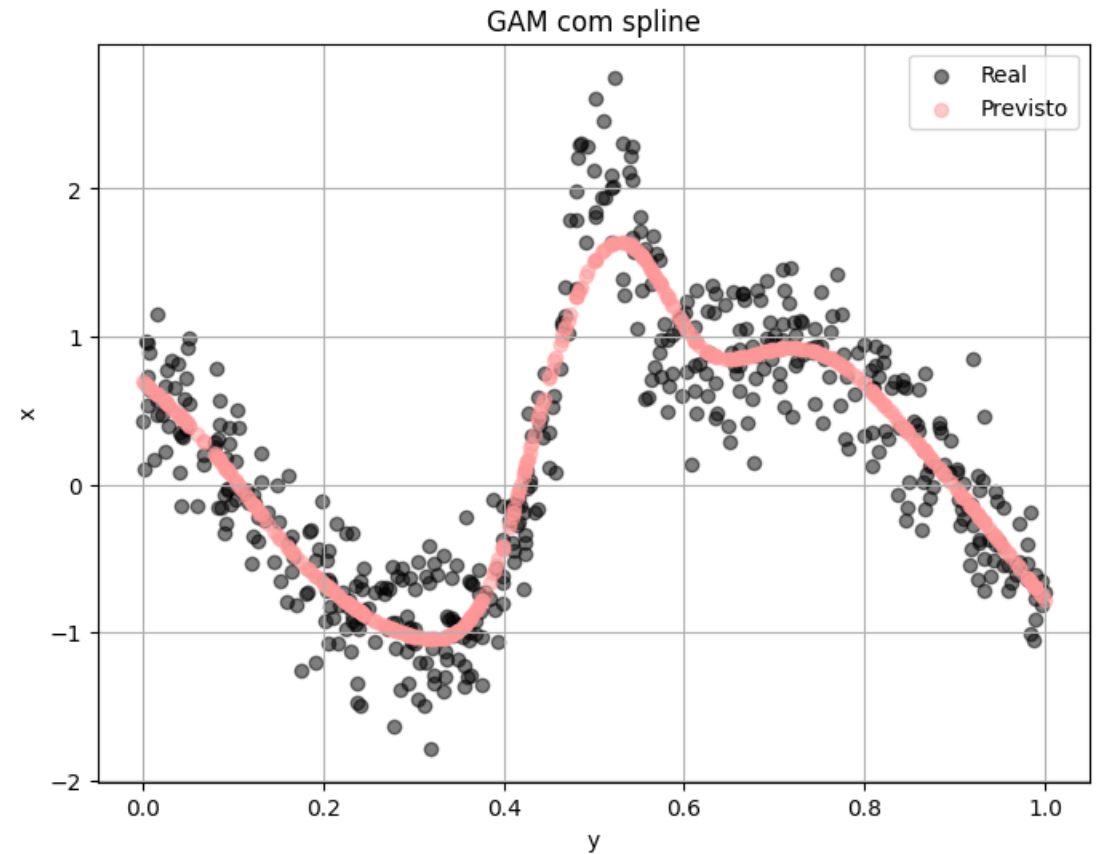
y_pred = modelo.predict(X)

plt.figure(figsize=(8, 6))
plt.scatter(X, y, color='black', alpha=0.5)
plt.scatter(X, y_pred, color='#FF9999', alpha=0.5)
plt.title("GAM com spline")
plt.xlabel("y")
plt.ylabel("x")
plt.grid(True)
plt.show()
```

```
print("Coeficientes", modelo.coef_)
print("Quantidade coeficientes", len(modelo.coef_))
```

✓ 0.0s

```
Coeficientes [ 0.93360326  0.57064711  0.11847733 -0.55832888 -0.99033304 -1.22763869
 -1.16223178  1.01894213  1.8357214   0.55285941  0.80869567  0.77482436
  0.2311301  -0.31070271 -0.91746293 -1.5346375   0.14356554]
Quantidade coeficientes 17
```



Exemplo com código Python

- Hiperparâmetros pyGAM
 - **Distribution**
 - Normal
 - Dados contínuos que seguem uma distribuição normal
 - Gamma
 - Dados contínuos positivos com uma distribuição assimétrica
 - Inv_gauss
 - Dados contínuos positivos com uma distribuição de cauda longa
 - **Link function**
 - Função de link que conecta a variável dependente (target) ao preditor
 - Identity
 - Logit
 - Inverse
 - Log
 - Inverse-squared

Exemplo com código Python


- Hiperparâmetros pyGAM
 - **Link function**
 - Exemplo para link igual a 'Log'.

$X = [1, 2, 3, 4, 5]$
 $y = [2, 3, 6, 8, 11]$

Modelo linear: $\eta = \beta_0 + \beta_1 X_n, n \in [1, 5]$.

Onde, $\beta_0 = 0.51$ e $\beta_1 = 0.38$

Para $X = 1, \eta = 0.51 + 0.38 * 1 = 0.89$
Para $X = 2, \eta = 0.51 + 0.38 * 2 = 1.27$
Para $X = 3, \eta = 0.51 + 0.38 * 3 = 1.65$
Para $X = 4, \eta = 0.51 + 0.38 * 4 = 2.03$
Para $X = 5, \eta = 0.51 + 0.38 * 5 = 2.41$

Link = 'log' 

Para $\eta = 0.89, \hat{y} = e^{0.89} = 2.43$
Para $\eta = 1.27, \hat{y} = e^{1.27} = 3.56$
Para $\eta = 1.65, \hat{y} = e^{1.65} = 5.21$
Para $\eta = 2.03, \hat{y} = e^{2.03} = 7.61$
Para $\eta = 2.41, \hat{y} = e^{2.41} = 11.13$

Exemplo com código Python

- **Exemplo com mais de uma variável**
 - Aluguel de bicicletas
 - Target é 'cnt' (quantidade de bicicletas alugadas no dia)

```
df_data
✓ 0.0s
```

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	2011-01-01	1	0	1	0	6	0	2	8.175849	7.999250	0.805833	0.160446	331	654	985
1	2	2011-01-02	1	0	1	0	0	0	2	9.083466	7.346774	0.696087	0.248539	131	670	801
2	3	2011-01-03	1	0	1	0	1	1	1	1.229108	-3.499270	0.437273	0.248309	120	1229	1349
3	4	2011-01-04	1	0	1	0	2	1	1	1.400000	-1.999948	0.590435	0.160296	108	1454	1562
4	5	2011-01-05	1	0	1	0	3	1	1	2.666979	-0.868180	0.436957	0.186900	82	1518	1600
...
726	727	2012-12-27	1	1	12	0	4	1	2	3.945849	-1.041628	0.652917	0.350133	247	1867	2114
727	728	2012-12-28	1	1	12	0	5	1	2	3.906651	0.833036	0.590000	0.155471	644	2451	3095
728	729	2012-12-29	1	1	12	0	6	0	2	3.906651	-0.001600	0.752917	0.124383	159	1182	1341
729	730	2012-12-30	1	1	12	0	0	0	1	4.024151	-0.707800	0.483333	0.350754	364	1432	1796
730	731	2012-12-31	1	1	12	0	1	1	2	2.144151	-1.249858	0.577500	0.154846	439	2290	2729

731 rows × 16 columns

Exemplo com código Python

- Entrada do modelo: sensação térmica

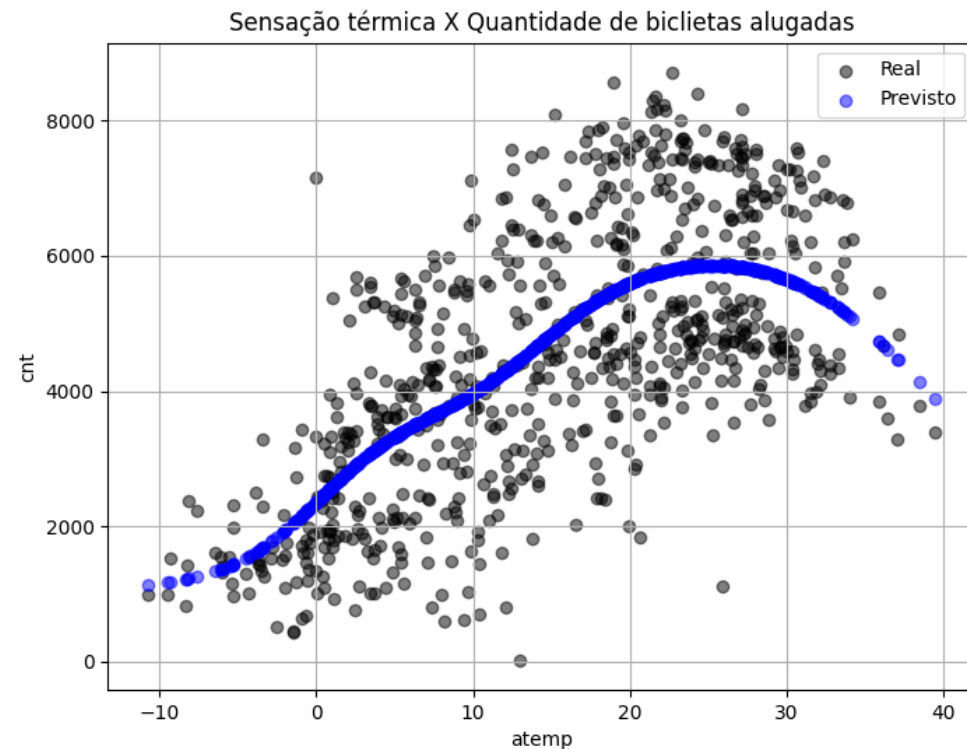
```
modelo = GAM(s(0, n_splines=10))

# Ajustando o modelo aos dados
modelo.fit(df_data["atemp"].values, df_data["cnt"].values)

df_data["prev"] = modelo.predict(df_data["atemp"].values)

plt.figure(figsize=(8, 6))
plt.scatter(df_data["atemp"].values, df_data["cnt"].values, color='black', alpha=0.5)
plt.scatter(df_data["atemp"].values, df_data["prev"].values, color='blue', alpha=0.5)
plt.title("Sensação térmica X Quantidade de bicicletas alugadas")
plt.xlabel("atemp")
plt.ylabel("cnt")

plt.grid(True)
plt.show()
```



Exemplo com código Python

- Entrada do modelo: sensação térmica, mês, flag de feriado, dia da semana, flag de dia útil e umidade

```
model_input = ["atemp", "mnth", "holiday", "weekday", "workingday", "hum"]
```

✓ 0.0s

```
['atemp', 'mnth', 'holiday', 'weekday', 'workingday', 'hum']
```

```
modelo = GAM(s(0, n_splines=5) + f(1) + f(2) + f(3) + f(4) + s(5, n_splines=5), link="log")
```

```
modelo.fit(df_data[model_input].values, df_data["cnt"].values)
```

```
df_data["prev"] = modelo.predict(df_data[model_input].values)
```

```
plt.figure(figsize=(8, 6))
plt.scatter(df_data["atemp"].values, df_data["cnt"].values, color='black', alpha=0.5)
plt.scatter(df_data["atemp"].values, df_data["prev"].values, color='blue', alpha=0.5)
plt.title("Sensação térmica X Quantidade de bicicletas alugadas")
plt.xlabel("atemp")
plt.ylabel("cnt")
plt.grid(True)
plt.show()
```

```
a = len(df_data["mnth"].unique())
b = len(df_data["holiday"].unique())
c = len(df_data["weekday"].unique())
d = len(df_data["workingday"].unique())
```

```
print("Quantidade de meses", a)
print("Identificador de feriado", b)
print("Quantidade de dias da semana", c)
print("Identificador de dias úteis", d)
total = a + b + c + d + 2*5 + 1
print("")
print("Total de coeficientes", total)
```

✓ 0.0s

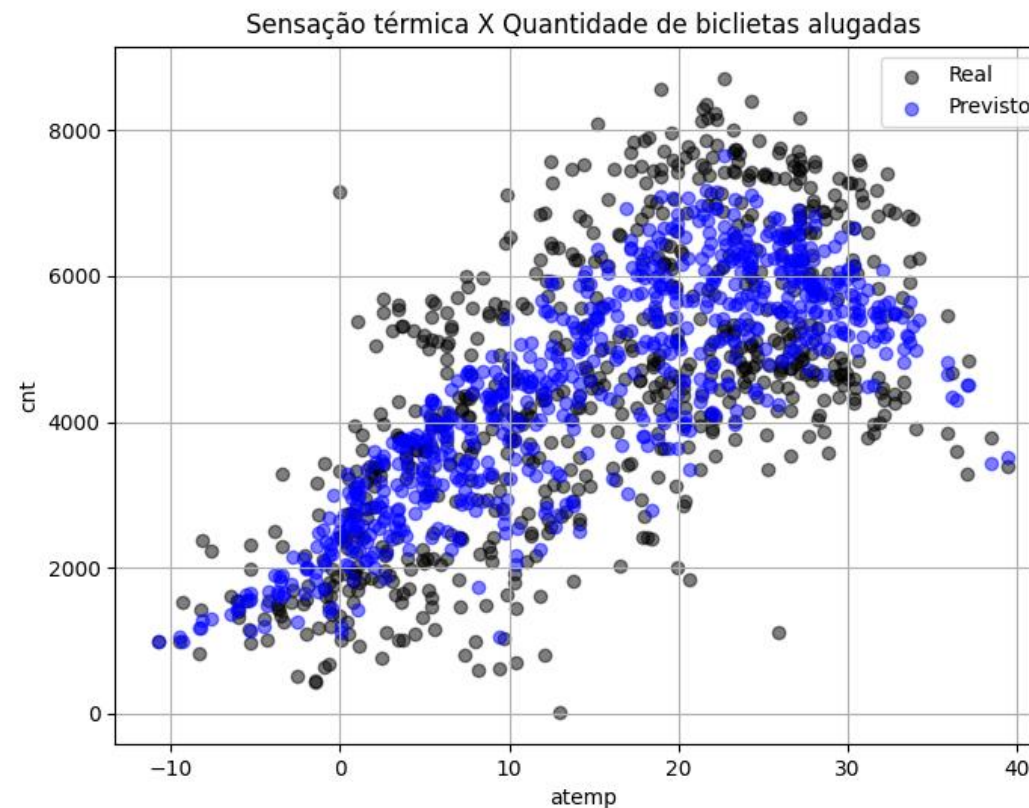
Quantidade de meses 12
Identificador de feriado 2
Quantidade de dias da semana 7
Identificador de dias úteis 2

Total de coeficientes 34

```
len(modelo.coef_)
```

✓ 0.0s

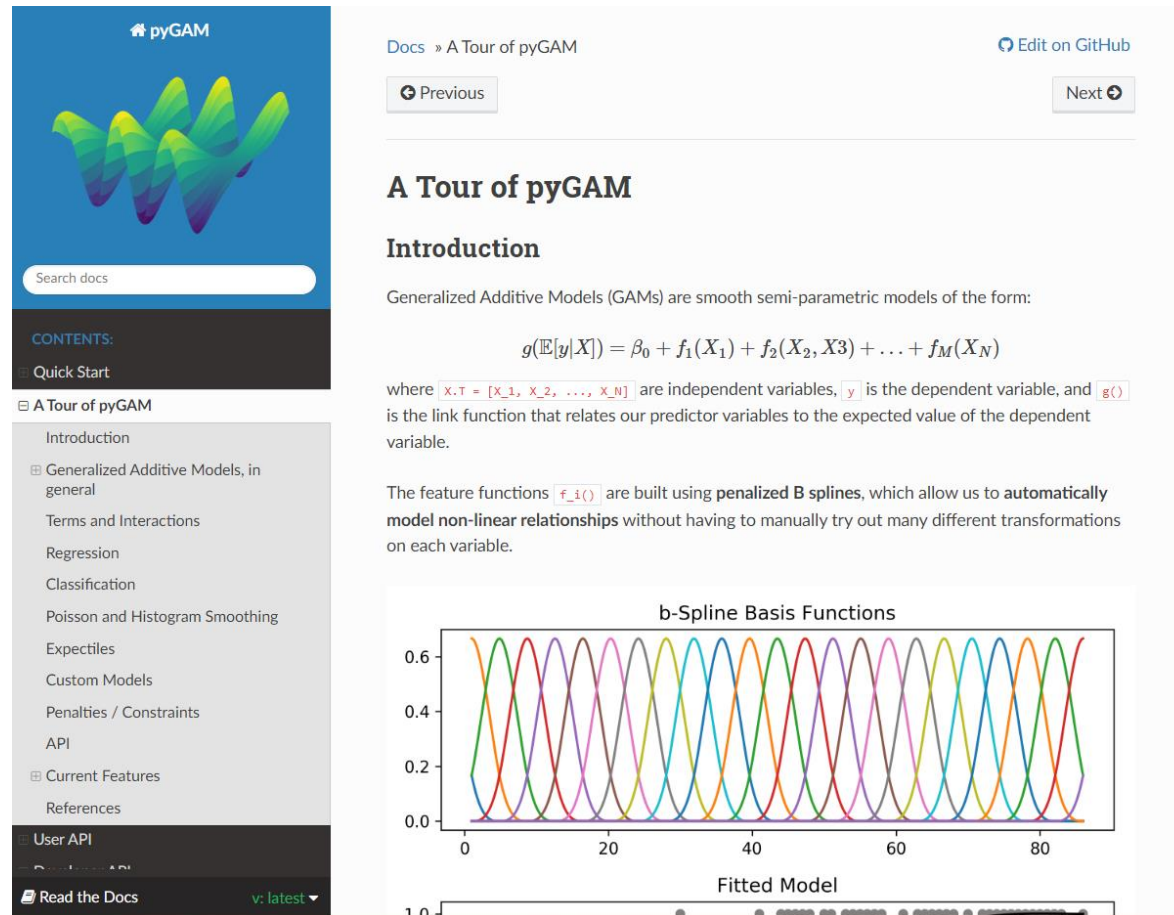
34



Vantagens e desvantagens

- Vantagens
 - Flexibilidade
 - Interpretabilidade
 - Lida bem com variáveis categóricas
- Desvantagens
 - Quantidade de coeficientes pode ser grande
 - Necessidade de especificação adequada

[Código no GitHub](#)



The screenshot displays the pyGAM documentation interface. On the left is a sidebar with a search bar and a table of contents including sections like 'Introduction', 'Generalized Additive Models, in general', 'Terms and Interactions', 'Regression', 'Classification', 'Poisson and Histogram Smoothing', 'Expectiles', 'Custom Models', 'Penalties / Constraints', 'API', 'Current Features', and 'References'. The main content area is titled 'A Tour of pyGAM' and includes an 'Introduction' section. It defines Generalized Additive Models (GAMs) as smooth semi-parametric models of the form:

$$g(\mathbb{E}[y|X]) = \beta_0 + f_1(X_1) + f_2(X_2, X_3) + \dots + f_M(X_N)$$

where $X.T = [X_1, X_2, \dots, X_N]$ are independent variables, y is the dependent variable, and $g(\cdot)$ is the link function that relates our predictor variables to the expected value of the dependent variable.

The text further explains that feature functions $f_i(\cdot)$ are built using penalized B splines, which allow for automatically modeling non-linear relationships without manual transformations.

Below the text is a plot titled 'b-Spline Basis Functions' showing multiple overlapping bell-shaped curves of different colors. At the bottom, a 'Fitted Model' plot shows a horizontal line with several data points.

Referências

- PyGAM Documentation. (2024). PyGAM: Generalized Additive Models in Python. Acesso em https://pygam.readthedocs.io/en/latest/notebooks/tour_of_pygam.html
- Clark, M. (2022). Generalized Additive Models. Acesso em <https://m-clark.github.io/generalized-additive-models/introduction.html>
- UFABC. (2022). Generalized Additive Models (GAM). [Vídeo]. YouTube. Acesso em https://www.youtube.com/watch?v=QQl4ixKPj2s&ab_channel=ufabc_hal
- UFSJ. (2022). Modelos Aditivos Generalizados (GAM) - Aula 1. [Vídeo]. YouTube. Acesso em https://www.youtube.com/watch?v=VmD0oDer_kM&ab_channel=CanaldaEngenhariadeManufaturaeQualidade
- Jackson, S. S. (2022). Generalised Additive Models. In Machine Learning Lecture Notes. Acesso em <https://bookdown.org/ssjackson300/Machine-Learning-Lecture-Notes/generalised-additive-models.html>
- Xiao, T. (2022). Introduction to PyGAM. Acesso em <https://tesixiao.github.io/teaching/2022-winter-142a/pyGAM>
- Hastie, T., & Tibshirani, R. (1986). Generalized Additive Models. Statistical Science, 1(3), 297-318. Acesso em <https://projecteuclid.org/journals/statistical-science/volume-1/issue-3/Generalized-Additive-Models/10.1214/ss/1177013604.full>

Obrigado!
marcelov.aguiar@gmail.com