Viés Variância

October 22, 2022

1 Erro de Regressão

Para saber se um ajuste está adequado, precisamos de um parametro para medir sua qualidade.

Nos modelos de regressão isso pode ser feito utilizando o **Erro Médio Quadrático**:

1.0.1 $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$

Onde:

O valor predito é comparado com o resultado real.

Elevando a diferença ao quadrado, diferenças maiores causam um impacto quadratico maior no erro.

Os algoritmos de regressão minimizam este erro ajustando os parâmetros do modelo.

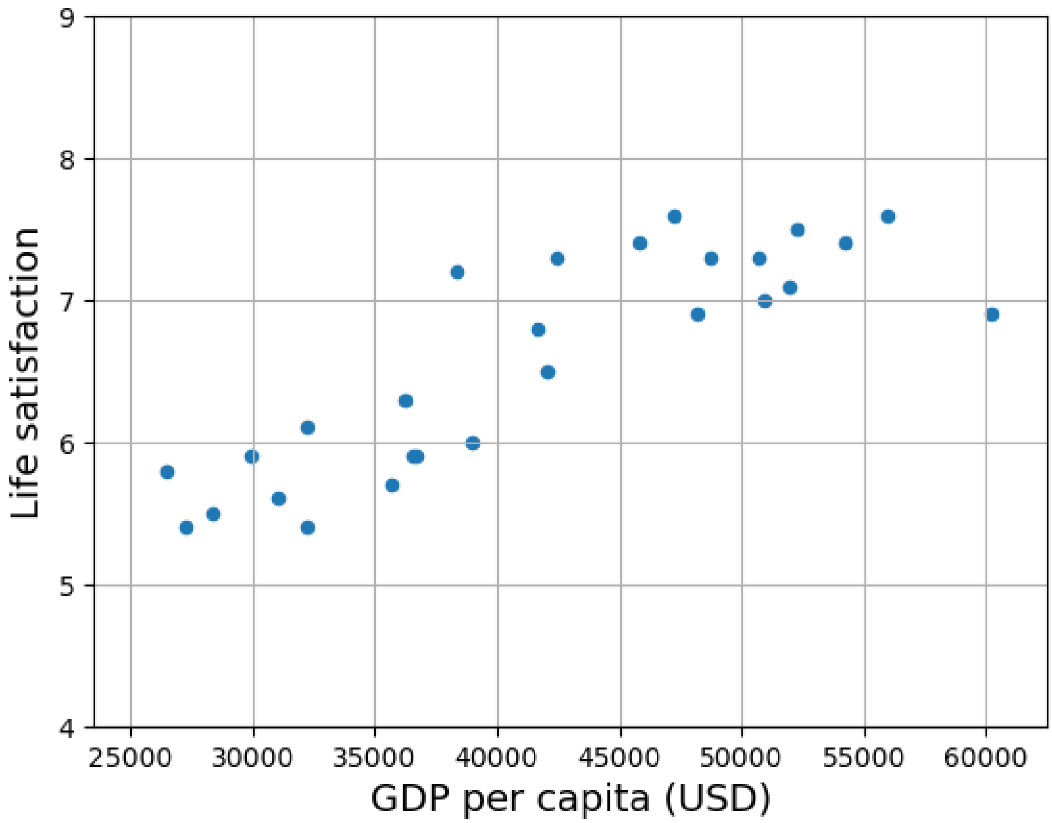
[2]:
```
from IPython.display import Image
%matplotlib inline
```

[3]:
```
import matplotlib.pyplot as plt

plt.rc('font', size=12)
plt.rc('axes', labelsize=14, titlesize=14)
plt.rc('legend', fontsize=12)
plt.rc('xtick', labelsize=10)
plt.rc('ytick', labelsize=10)
```

[4]:
```
import numpy as np

np.random.seed(42)
```

[5]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression

# Baixando e preparando os dados
lifesat = pd.read_csv("lifesat.csv")
X = lifesat[["GDP per capita (USD)"]].values
```

```python
y = lifesat[["Life satisfaction"]].values

# Visualizando os dados the data
lifesat.plot(kind='scatter', grid=True,
             x="GDP per capita (USD)", y="Life satisfaction")
plt.axis([23_500, 62_500, 4, 9])
plt.show()

# Selecione o modelo de regressão linear
model = LinearRegression()

# Treine o modelo
model.fit(X, y)

# Fazer uma previsão para o Chipre
X_new = [[37_655.2]]  # PIB per capita de Chipre em 2020
print(model.predict(X_new)) # saida [[6.30165767]]
```




```
[[6.30165767]]
```

Substituindo o modelo d regressão linear por k-vizinhos mais próximos (neste exemplo para k=3)

[5]:
```python
from sklearn.neighbors import KNeighborsRegressor

model = KNeighborsRegressor(n_neighbors=3)

model.fit(X, y)

print(model.predict(X_new))
```

```
[[6.33333333]]
```

[8]:
```python
import urllib.request
from pathlib import Path
datapath = Path() / "datasets" / "lifesat"
datapath.mkdir(parents=True, exist_ok=True)
```

[9]:
```python
oecd_bli = pd.read_csv(datapath / "oecd_bli.csv")
gdp_per_capita = pd.read_csv(datapath / "gdp_per_capita.csv")
```

[10]:
```python
gdp_year = 2020
gdppc_col = "GDP per capita (USD)"
lifesat_col = "Life satisfaction"
gdp_per_capita = gdp_per_capita[gdp_per_capita["Year"] == gdp_year]
gdp_per_capita = gdp_per_capita.drop(["Code", "Year"], axis=1)
gdp_per_capita.columns = ["Country", gdppc_col]
gdp_per_capita.set_index("Country", inplace=True)
gdp_per_capita.head()
```

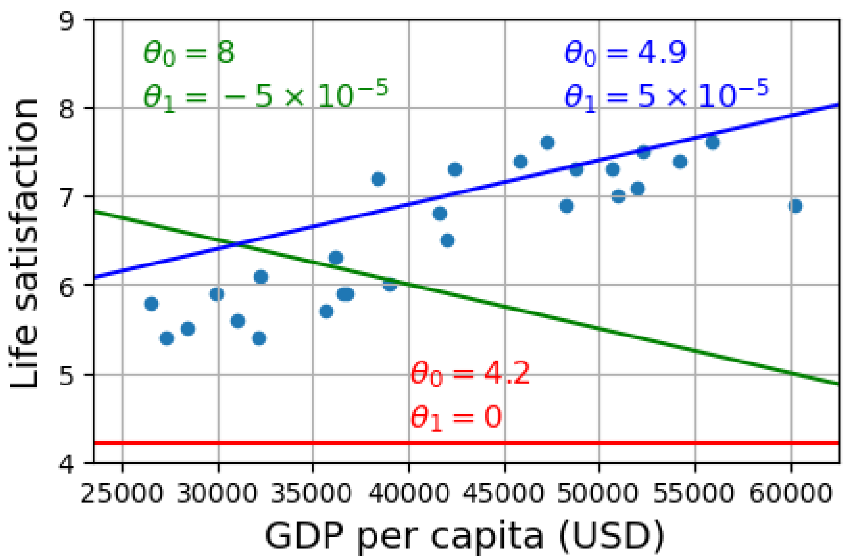
[10]:

	GDP per capita (USD)
Country	
Afghanistan	1978.961579
Africa Eastern and Southern	3387.594670
Africa Western and Central	4003.158913
Albania	13295.410885
Algeria	10681.679297

[11]:
```python
min_gdp = 23_500
max_gdp = 62_500
```

[12]:
```python
lifesat.plot(kind='scatter', figsize=(5, 3), grid=True,
x=gdppc_col, y=lifesat_col)
X = np.linspace(min_gdp, max_gdp, 1000)
min_life_sat = 4
max_life_sat = 9
w1, w2 = 4.2, 0
plt.plot(X, w1 + w2 * 1e-5 * X, "r")
plt.text(40_000, 4.9, fr"$\theta_0 = {w1}$", color="r")
plt.text(40_000, 4.4, fr"$\theta_1 = {w2}$", color="r")
w1, w2 = 8, -5
```

```python
plt.plot(X, w1 + w2 * 1e-5 * X, "g")
plt.text(26_000, 8.5, fr"$\theta_0 = {w1}$", color="g")
plt.text(26_000, 8.0, fr"$\theta_1 = {w2} \times 10^{{-5}}$", color="g")
w1, w2 = 4.9, 5
plt.plot(X, w1 + w2 * 1e-5 * X, "b")
plt.text(48_000, 8.5, fr"$\theta_0 = {w1}$", color="b")
plt.text(48_000, 8.0, fr"$\theta_1 = {w2} \times 10^{{-5}}$", color="b")
plt.axis([min_gdp, max_gdp, min_life_sat, max_life_sat])
#save_fig('tweaking_model_params_plot')
plt.show()
```



θ_0 = Intercepta a origem θ_1 = inclinação da linha

Para cada predição do modelo, compara com o valor y_i, que é o real

Diferenças maiores causam impactos maiores

2 Equlibrio entre Viés Variância

Existem três tipos de erros:

- **viés** = erro de aproximação causado pelo modelo;
- **variância do modelo** = o quanto a função f variaria se mudássemos o conjunto de dados selecionado;
- **variância do ruído** = erro causado por ruído nos dados

2.1 $E(y-\hat{f}(x))^2 = Var(\hat{f}(x)) + [Bias(\hat{f}(x))]^2 + Var(\epsilon)$

2.2 $E(y-\hat{f}(x))^2 = E[\hat{f}(x) - E[\hat{f}(x)]]^2 + [E[\hat{f}(x)] - f(x)]^2 + Var(\epsilon)$

Onde:

- $\hat{f}(x)$ = valor predito
- $\hat{f}(x)$ = valor real
- $E[\hat{f}(x)]$ = média dos valores preditos

$Bias^2$ = Quanto os valores preditos diferem dos valores reais

$Var(\hat{f}(x))$ = Quanto as predições realizadas com os mesmos valores variam em diferentes realizações do modelo.

Muitas vezes, usamos o termo equilibrio entre viés e variância ou comprensação entre viés e variância para descrever o desempenho de um modelo

- **alta variância** é proporcional ao sobreajuste (overfitting) = origina-se de proposições complexas para o modelo
- **alto viés** é proporcional ao subajuste (underfitting) = origina-se de proposições simplificadas para o modelo.

O viés é comumente definido como a diferença entre o valor esperado do estimador e o parâmetro que queremos estimar

$Bias = E[\hat{f}(x)] - f(x)$

$Var(\hat{f}(x)) = E[(\hat{f}(x))^2] - (E[\hat{f}(x)])^2 = E[(E[\hat{f}(x)] - \hat{f}(x))]^2$

2.3 Equações de viés-variância

$E[(\hat{\theta}-\theta)^2] = E[(\hat{\theta} - E[\hat{\theta}])^2] + (E[\hat{\theta}-\theta)^2]$

MSE = Variância + Viés^2

Onde

MSE = Erro Médio ao Quadrado

θ = parâmetro do modelo, que corresponde á variável alvo representado pela letra y.

Assim, temos que:

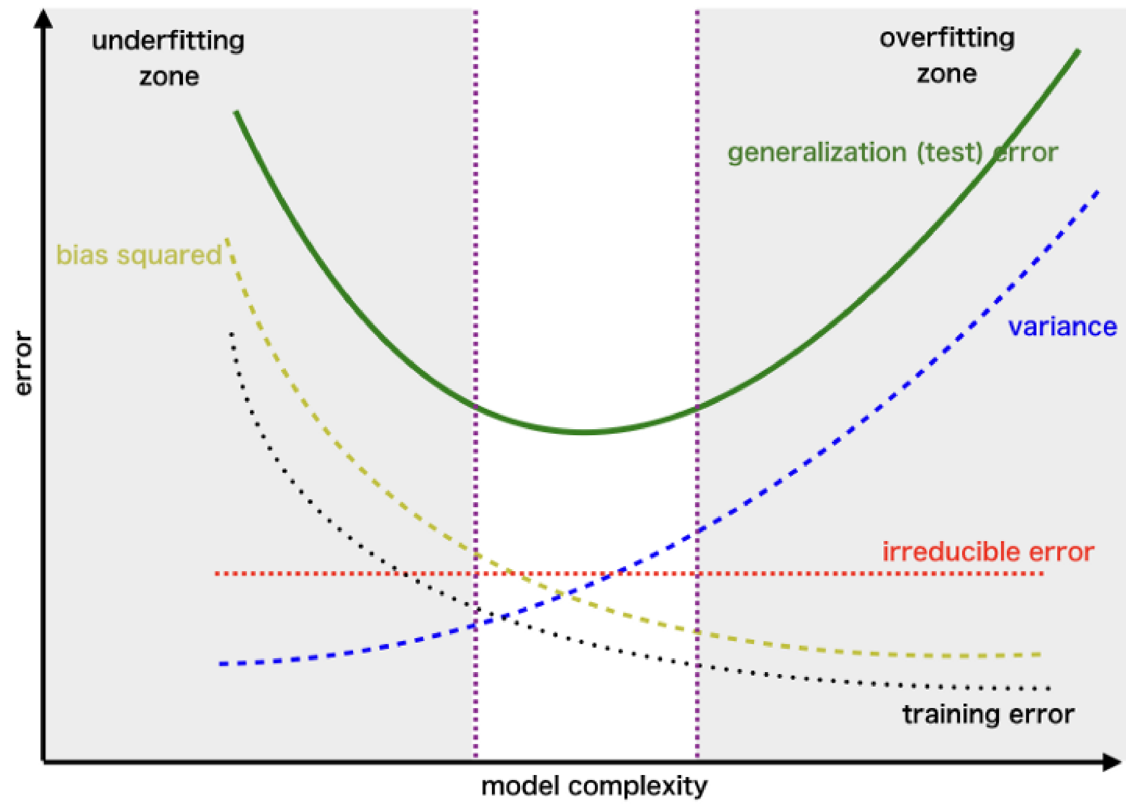
$E[(\hat{y}-y)^2] = E[(\hat{y} - E[\hat{y}])^2] + (E[\hat{y}-y)^2]$

Onde:

- y = valor real da variável alvo
- $\hat{y}$ = predição da variável alvo
- $E[\hat{y}]$ = Esperança da predição da variável alvo

[6]:
```
Image(filename='1_oOOKYF7Z84nePqfsJ9EOWQ.png')
```

[6]:



Essa decomposição é usada para exlicar como o resultado de um modelo muda baseado em sua complexidade:

Baixa Complexidade: Os modelos estimados tendem a ser semelhantes dessa forma a **variância é pequena**. Por outro lado o modelo médio não é poderoso o suficiente para se aproximar do resultado real, dessa forma o **viés é grande**.

Alta Complexidade: Os modelos estimados tendem a ser diferentes dessa forma a **variância é alta**. Por outro lado o modelo médio torna-se expecializado de mais se aproximando muito do resultado real, dessa forma o **viés é muito pequeno**.

2.4 Tradeoff Viés Variância na Prática

Suponha que a função verdade que devá ser maximizada seja a função seguinte:

$f(x) = \frac{1}{2}x + \sqrt{max(x, 0} - \cos x + 2$

E o ruido seja modelado como uma Gaussiana com média zero e desvio padrão 1, tal que:

$\epsilon = N(0, 1)$

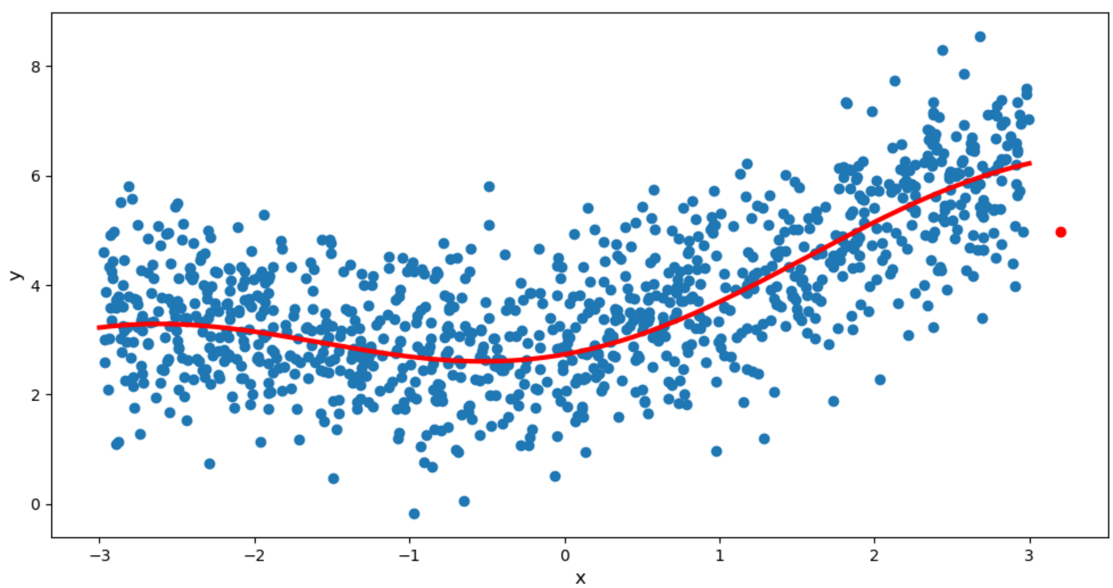
Assim, temos que:

$y = f(x) + \epsilon$

Se gerarmos 100 pontos a partir desse processo obtemos:

[7]:

```python
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
%matplotlib inline

def f(x):
    return .5 * x + np.sqrt(np.max(x, 0)) - np.cos(x) + 2

N = 1000
sigma_epsilon = 1
x_max = 3
x_test = 3.2
x = x_max * (2 * np.random.rand(N) - 1)
epsilon = sigma_epsilon * np.random.randn(N)

y = f(x) + epsilon
y_test = f(x_test) + sigma_epsilon * np.random.randn()

plt.figure(figsize=(12, 6))
x_range = np.linspace(-x_max, x_max, 1000)
plt.scatter(x, y)
plt.plot(x_range, f(x_range), 'r', linewidth=3.0)
plt.scatter(x_test, y_test, c='r')
plt.xlabel('x', size=12)
plt.ylabel('y', size=12)
plt.xticks(np.arange(-x_max, x_max + 1))
plt.show()
```



Os pontos azuis representam os parares x e y e a linha vermelha e a função $f(x)$.

O ponto em vermelho representa o ponto teste que queremos predizer.

É possível notar que $f(x)$ tem um comportamento não linear devido a presença da raiz quadrada e do cosseno na função.

2.4.1 Modelando utilizando regressões polinomiais

O problema será modelado utilizando regressões polinomiais de vários graus de complexidade.

A regressão polinimial tem por objetivo ajustar uma relação nào linear entre x e y com a seguinte função:

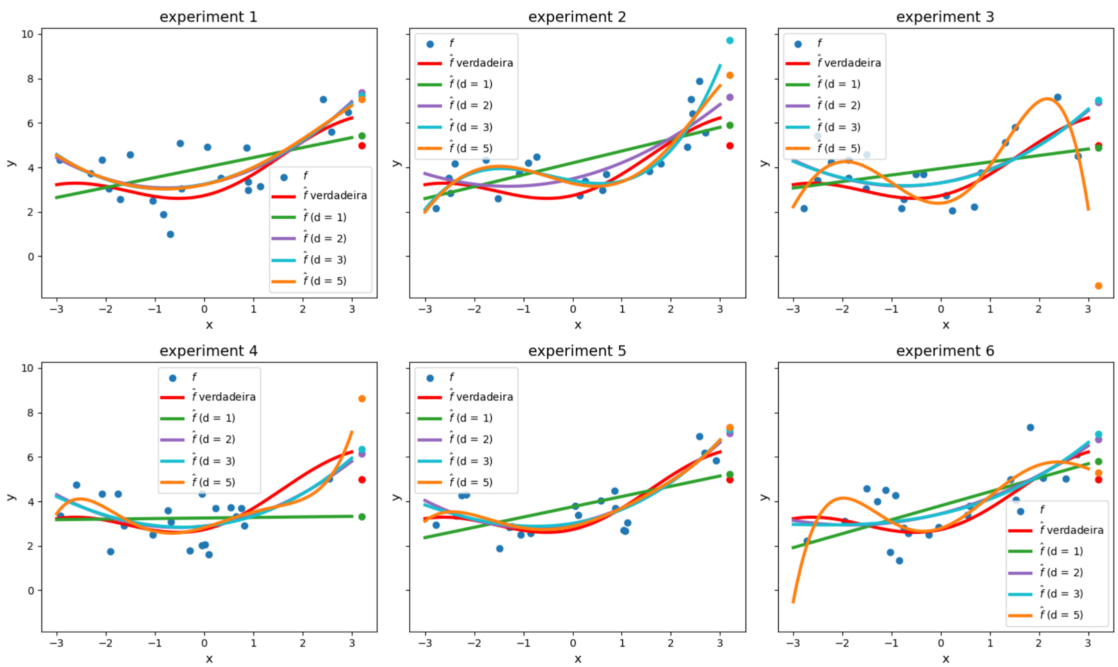
$\hat{f}(x) = w_0 + w_1 x + w_2 x^2 + \cdots + w_d x^d$

Para tanto vamos supor que dos 1000 pontos do exemplo anterior usaremos aleatóriamente 20 pontos que são instâncias distintas do conjunto de dados anterior e vamos treinar nossa regressão polinomial com 4 graus de regressão distintos que correspondem aos graus das funcões polinomiais usadas. Usaremos os seguintes graus: 1, 2m 3 e 5.

Iremos repetir 6 vezes esse processo para obtermos 6 instâncias diferentes.

[8]:
```
def f_hat(x, w):
    d = len(w) - 1
    return np.sum(w * np.power(x, np.expand_dims(np.arange(d, -1, -1), 1)).T, 1)

n = int(.02 * N)
x_test = 3.2
x_range = np.linspace(-x_max, x_max, 1000)
colors = np.array(['tab:green', 'tab:purple', 'tab:cyan', 'tab:orange'])
d_arr = [1, 2, 3, 5]

cnt = 1
fig, axs = plt.subplots(2, 3, sharey=True, figsize=(15, 9))
for i in range(2):
    for j in range(3):
        idx = np.random.permutation(N)[:n]
        x_train, y_train = x[idx], y[idx]

        w = []
        for d in d_arr:
            w.append(np.polyfit(x_train, y_train, d))

        axs[i, j].scatter(x_train, y_train)
        axs[i, j].plot(x_range, f(x_range), 'r', linewidth=3.0)
        for k in range(len(w)):
            axs[i, j].plot(x_range, f_hat(x_range, w[k]), colors[k],
 ↪linewidth=3.0)

        axs[i, j].scatter(x_test, y_test, c='r')
```

```
        for k in range(len(w)):
            axs[i, j].scatter(x_test, f_hat(x_test, w[k]), c=colors[k])

        axs[i, j].set_xlabel('x', size=12)
        axs[i, j].set_ylabel('y', size=12)
        axs[i, j].legend([r'$f$', r'$\hat{f}$ verdadeira', r'$\hat{f}$ (d = 1)',
                          r'$\hat{f}$ (d = 2)', r'$\hat{f}$ (d = 3)', r'$\hat{f}$ (d = 5)'], fontsize=10)
        axs[i, j].title.set_text('experiment {}'.format(cnt))
        cnt += 1
plt.tight_layout()
plt.show()
```



Os pontos azuis representam os 20 pontos.

A linha vermelha é a função obtida no gráfico anterior por regressão linear.

Os pontos verde, roxo, azul claro e laranja representam a previsão do valor de x para cada regressão polinimial.

É possível observar que temos menos variação no comportamento das linhas com menor grau de complexidade.

Em $d = 1$ em verde corresponde a uma função de primeiro grau, que não muda muito com oa variação dos dados de treinamento.

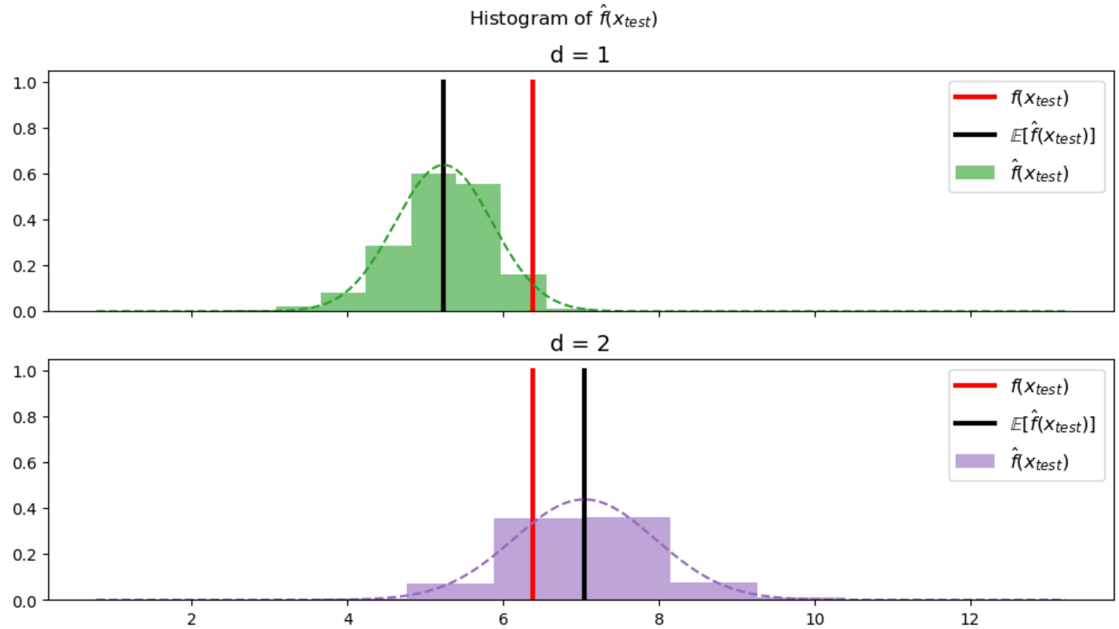
Já a linha $d = 5$ em laranja é mais sensivel a pequenas flutuações nos dados de treinamento.

Esse é o problema da variância, onde um modelo mais simples torna-se mais robusto as mudanças dos dados de treinamento, contudo o viés, que corresponde a média de $f(x)$, é maior para modelos mais simples. Isso ocorre pois a função para $d = 1$ nào é tão representativa quanto a função verdadeira.

Smulando 10000 experimentos diferentes, aleatórios buscando 20 pontos da população, para cada experimento dado um valor de x, vamos obter sua resultante $\hat{f}(x)$, assim teremos 10 mil previsões de $\hat{f}(x)$. Faremos isso para os dois mdelos de regressão de grau 1 (d=1) e grau 2 (d=2).

Com esses dados obteremos a média dos valores $E[\hat{f}(x)]$, a média do resultado real f(x) e os valores preditos $\hat{f}(x)$ em um histograma.

[9]:
```
R = 10000
d_arr = [1, 2, 3, 5]
y_hat_test = np.zeros((len(d_arr), R))

for r in range(R):
    n = int(.02 * N)
    idx = np.random.permutation(N)[:n]
    x_train, y_train = x[idx], y[idx]

    for k in range(len(d_arr)):
        d = d_arr[k]
        w = np.polyfit(x_train, y_train, d)
        y_hat_test[k, r] = f_hat(x_test, w)

y_hat_test_mean = np.mean(y_hat_test, 1)
y_hat_test_std = np.std(y_hat_test, 1)

fig, axs = plt.subplots(2, 1, sharex=True, sharey=True, figsize=(12, 6))
for k in range(2):
    axs[k].hist(y_hat_test[k], density=True, color=colors[k], alpha=0.6)
    xlim = axs[k].get_xlim()
    axs[k].plot([f(x_test), f(x_test)], [0, 1], 'r', linewidth=3.0)
    axs[k].plot([y_hat_test_mean[k], y_hat_test_mean[k]], [0, 1], c='k',␣
 ↪linewidth=3.0)
    axs[k].title.set_text('d = {}'.format(d_arr[k]))
    axs[k].legend([r'$f(x_{test})$', r'$\mathbb{E}[\hat{f}(x_{test})]$',␣
 ↪r'$\hat{f}(x_{test})$'], fontsize=12)

for k in range(2):
    x_range = np.linspace(xlim[0], xlim[1], 1000)
    axs[k].plot(x_range, stats.norm.pdf(x_range, y_hat_test_mean[k],␣
 ↪y_hat_test_std[k]), color=colors[k], ls='--')

plt.suptitle(r'Histogram of $\hat{f}(x_{test})$', size=12)
plt.show()
```



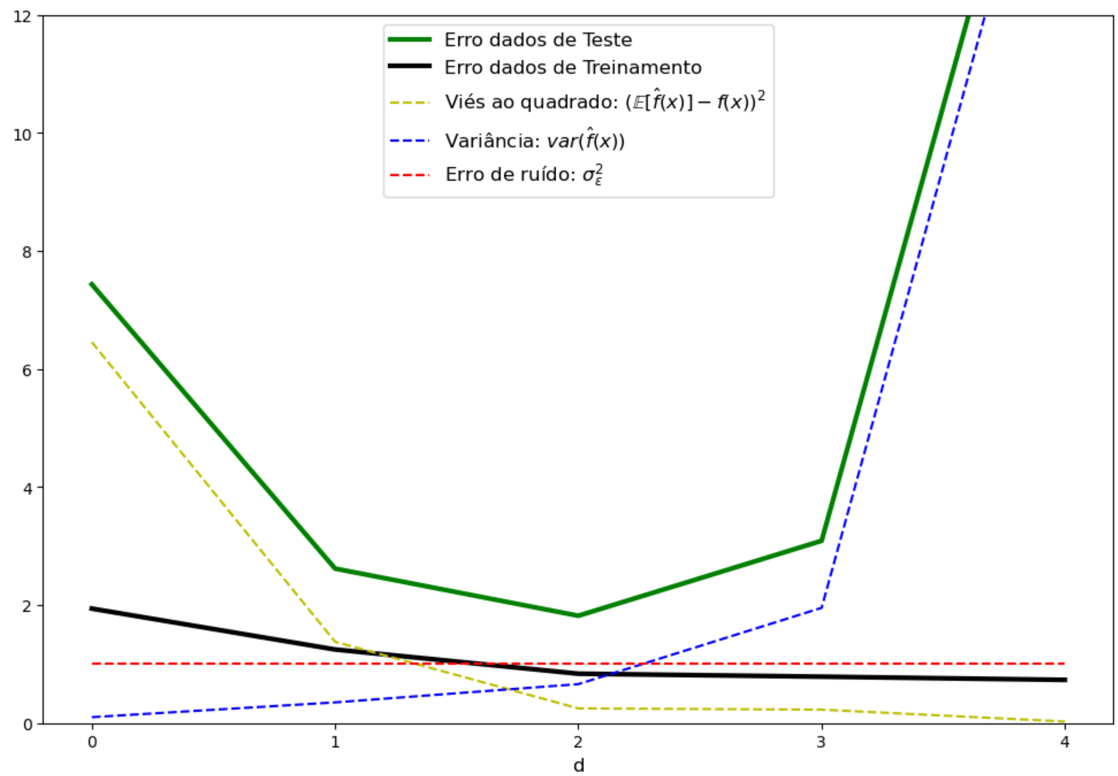
A média de $\hat{f}(x)$ representada pela linha preta está mais longe da média dos valores verdadeiros $f(x)$, representado pela linha em vermelho. Comparando as funções de primero e segundo grau.

Essa diferença entre as médias das funções de primeiro e segundo grau, representa o viés, ou seja o desvio entre os resultados do modelo e o valor real, causado pela simplicidade do modelo.

Por outro lado a variância é maior no modelo quadrático em comparação ao modelo linear. Isso pode ser observado pela maior dispersão do histograma em roxo.

[12]:
```
R = 10000
n = int(.02 * N)
n_test = 1000
d_arr = np.arange(5)

x_test = x_max + np.random.rand(n_test) - .5
epsilon = sigma_epsilon * np.random.randn(n_test)
y_test = f(x_test) + epsilon

train_squared_error = np.zeros((len(d_arr), R))
y_hat_test = np.zeros((len(d_arr), R, n_test))
for r in range(R):
    n = int(.02 * N)
    idx = np.random.permutation(N)[:n]
    x_train, y_train = x[idx], y[idx]
    for k in range(len(d_arr)):
        d = d_arr[k]
        w = np.polyfit(x_train, y_train, d)
```

```
        train_squared_error[k, r] = np.mean((y_train - f_hat(x_train, w)) ** 2)
        y_hat_test[k, r, :] = f_hat(x_test, w)

test_squared_error = np.mean((y_hat_test - y_test) ** 2, 1)
bias_squared = (np.mean(y_hat_test, 1) - f(x_test)) ** 2
var_y_hat_test = np.var(y_hat_test, 1)

plt.figure(figsize=(12, 8))
plt.plot(d_arr, np.mean(test_squared_error, 1), 'g', linewidth=3.0)
plt.plot(d_arr, np.mean(train_squared_error, 1), 'k', linewidth=3.0)
plt.plot(d_arr, np.mean(bias_squared, 1), 'y--')
plt.plot(d_arr, np.mean(var_y_hat_test, 1), 'b--')
plt.plot(d_arr, (sigma_epsilon ** 2) * np.ones_like(d_arr), 'r--')

plt.xticks(d_arr)
plt.xlabel('d', size=12)
plt.legend(['Erro dados de Teste', 'Erro dados de Treinamento', r'Viés ao␣
 ↪quadrado: $(\mathbb{E}[\hat{f}(x)] - f(x))^2$',
           r'Variância: $var(\hat{f}(x))$', r'Erro de ruído:␣
 ↪$\sigma_\epsilon^2$'], loc='upper center', fontsize=12)
plt.ylim([0, 12])
plt.show()
```



Somando as linhas amarela (viés ao quadrado), azul (variância) e vermelha (erros de ruído) temos a relação viés variância.

A linha preta representa o MSE = Erro Médio Quadrático, que reduz com a complexidade do modelo, visto que modelos mais complexos se ajustam melhor aos dados de treinamento.

O melhor resultado para o modelo corresponde ao modelo quadrático ($d = 2$)