Estimativa de Densidades Kernel
UFRGS - FIS01082

Aluno: Henrique Alexandre Boneto - 288744

Junho 2020

Função Densidade de Probabilidade (FDP)

Uma função densidade de probabilidade é definida como:

$$f_X(x) = \frac{dF_X(x)}{dx}$$

onde $F_X(x)$ é a função distribuição acumulada. Tal função nos dá a probabilidade de uma variável aleatória X ser menor ou igual a um valor x. A FDP por sua vez nos dá a probabilidade relativa de uma variável aleatória assumir um valor x.

A FDP tem as seguintes propriedades:

$$f_X \geq 0, \forall x \in \mathbb{R}$$

$$\int_{-\infty}^{\infty} f_x(u)du = 1$$

$$P(a < X \leq b) = F_x(b) - F_x(a) = \int_a^b f_X(u)du$$

Estimativa de Densidades Kernel

Quando não temos conhecimento sobre a densidade de um conjunto de dados, um dos métodos mais usados para encontrá-la é conhecida como estimativa de densidade kernel. Ela é dada por:

$$f(x) = \frac{1}{N}\sum_{n=1}^{N} K\Big(\frac{x - x_n}{2h}\Big) = \frac{1}{N}\sum_{n=1}^{N} K(u)$$

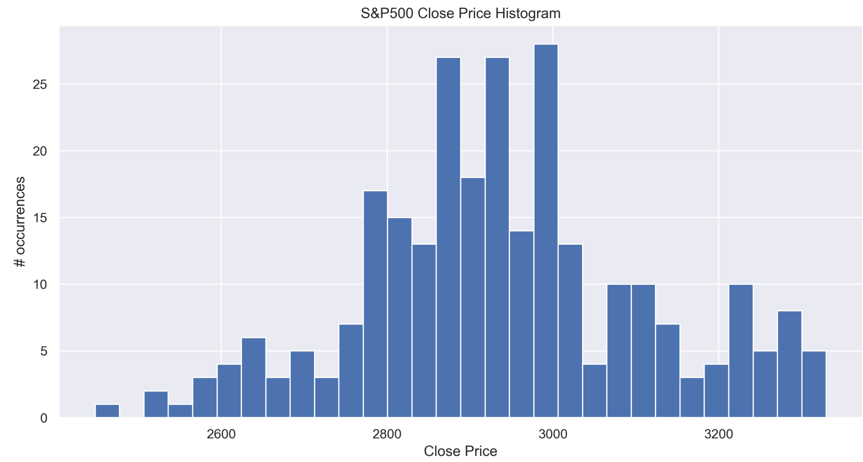
Onde $K(x)$ é função Kernel, h é um parâmetro de suavização e N é o tamanho da nossa série.

Exemplos de Kernel

Alguns dos mais kernels mais utilizados são:
- Kernel Uniforme (janela): $K(u) = \frac{1}{2}$, se $|u| \leq 1$
- Gaussiana: K(u) = $\frac{1}{\sqrt{2\sqrt{2\pi}}} \exp -\frac{1}{2}u^2$
- Kernel Uniforme (parabólico): $K(u) = \frac{3}{4}(1 - u^2)$, se $|u| \leq 1$

Série de Dados

Neste trabalho foram implementados os três exemplos de Kernel dados na seção anterior. A estimativa foi feita em cima dos dados históricos dos preços de fechamento do índice americano S&P500 entre o período de 2019-01-02 e 2020-01-31 obtidos pelo site Yahoo! Finance.
Abaixo o histograma da série de dados, com parâmetro "bins" igual a 30:



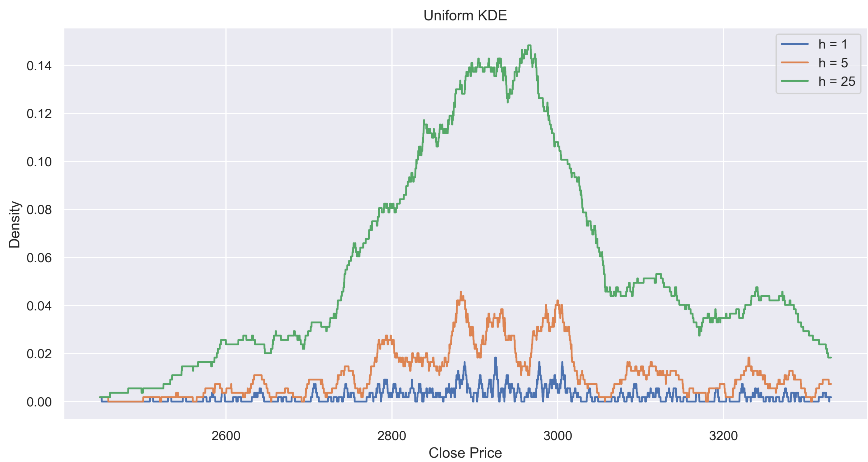
A partir desse gráfico é possível perceber que o preço de fechamento nesse período oscilou principalmente entre \$2800 e \$3100 aproximadamente, com um leve aumento após \$3200.

Resultados

Para checar a influência do parâmetro "h" nas estimativas, usou-se três valores diferentes, sendo eles os valores inteiros $1, 5$ e 25. O menor valor seria o de menor suavização, ou seja, aquele que muito mais facilmente se adqeua aos dados e possívelmente será classificado como "overfitting". Para avaliar se isso e é verdade e checar o quão bem os outros valores se sairão, fazemos os gráficos para cada um dos três kernels.

Kernel Uniforme

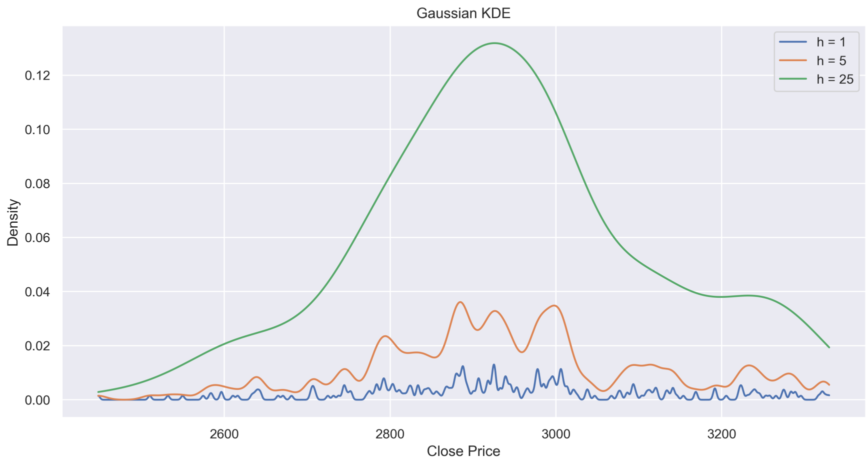
Na seção "Implementação" encontra-se sua implementação no método "uniform" dentro da classe "Kernel()". O gráfico obtido foi o seguinte:



Como esse kernel é feito a partir de janelas/caixas, ele tem o aspecto de ter menor suavização. Também é notório que para os valores de h=1 e h=5, a estimativa ficou extremamente influenciada pelos dados, com altos e baixos constantes, e assim, não seriam classificadas como boas.

Kernel Gaussiano

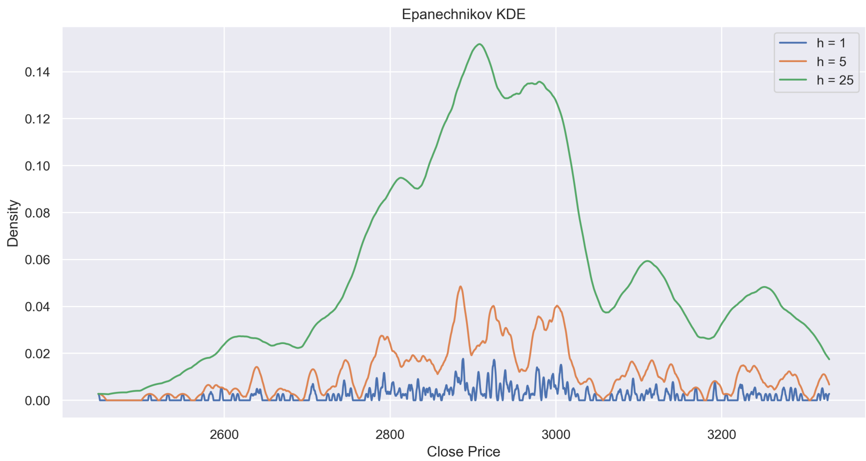
Também é possível encontrar a implementação desse Kernel na seção "Implementação". O gráfico obtido encontra-se abaixo:



Olhando para o resultado com h=25, a suavização dos pontos aumentou em comparação com o kernel anterior por conta da função escolhida. Alguns pequenos altos e baixos que existiam após os valores de preço de$3000 deixaram de existir. A conclusão para os valores de h=1 e h=5 permanece a mesma.

Kernel Epanechnikov

Por fim, o kernel Epanechnikov. Para ele, obteve-se o seguinte gráfico:



O resultado para tal kernel ficou similar com o kernel uniforme, apresentando os mesmo comportamentos com picos ressaltados, mas com maior suavização.

Implementação

Abaixo o código feito em Python:

```
#author: Henrique Boneto

#importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

class Kernel():
```

```
    """
    Some commons KDEs implementation in 1D
    """

    def __init__(self, x: list, range: float):
        self.xn = sorted(x)
        self.N = len(x)
        self.range = range
        self.X = np.arange(self.xn[0], self.xn[self.N - 1], self.range)

    def uniform(self, h):
        f = []
        for x in self.X:
            U = filter(lambda x: abs(x) <= 1, [(x - xn) / (2 * h) for xn in self.xn])
            F = [((1 / self.N) * (1 / 2)) for u in U]
            f.append(sum(F))

        return(self.X, f)

    def gaussian(self, h):
        f = []
        for x in self.X:
            U = [(x - xn) / (2 * h) for xn in self.xn]
            F = [((1 / self.N) * (1 / np.sqrt(2*np.pi)) *
                    (np.exp(-(1/2)*(u)**2))) for u in U]
            f.append(sum(F))

        return(self.X, f)

    def epanechnikov(self, h):
        f = []
        for x in self.X:
            U = filter(lambda x: abs(x) <= 1, [(x - xn) / (2 * h) for xn in self.xn])
            F = [((3 / 4) * (1 / self.N) * (1 - u**2)) for u in U]
            f.append(sum(F))

        return(self.X, f)

#data
df = pd.read_csv('data/S&P500.csv')
df = df.dropna()
x = list(df['Close'])

#plotting histogram
sns.set()
plt.figure(figsize=(12,6))
plt.hist(df['Close'], bins = 30)
plt.title('S&P500 Close Price Histogram')
plt.xlabel('Close Price')
plt.ylabel('# occurrences')
plt.savefig('figs/hist_close_S&P500.png', dpi=300)

#instantiation
kernel = Kernel(x, 0.1)

#definition of smoothing bandwidth "h"
hs = [1,5,25]
```

```
#plotting results
plt.figure(figsize=(12,6))
plt.title('Uniform KDE')
plt.xlabel('Close Price')
plt.ylabel('Density')
for h in hs:
    X, uniform = kernel.uniform(h)
    plt.plot(X, uniform, label='h = ' + str(h))
    plt.legend()
    plt.savefig('figs/uniform_kde.png', dpi=300)

plt.figure(figsize=(12,6))
plt.title('Gaussian KDE')
plt.xlabel('Close Price')
plt.ylabel('Density')
for h in hs:
    X, gaussian = kernel.gaussian(h)
    plt.plot(X, gaussian, label='h = ' + str(h))
    plt.legend()
    plt.savefig('figs/gaussian_kde.png', dpi=300)

plt.figure(figsize=(12,6))
plt.title('Epanechnikov KDE')
plt.xlabel('Close Price')
plt.ylabel('Density')
for h in hs:
    X, epanechnikov = kernel.epanechnikov(h)
    plt.plot(X, epanechnikov, label='h = ' + str(h))
    plt.legend()
    plt.savefig('figs/epanechnikov_kde.png', dpi=300)
```

Referências

[1] Probabilitycourse.com. 2020. Probability Density Function — PDF — Distributions. [online] Available at: https://www.probabilitycourse.com/chapter4/4_1_1_pdf.php [Accessed 6 June 2020].

[2] Faculty.washington.edu. 2020. [online] Available at: http://faculty.washington.edu/yenchic/18W_425/Lec6_hist_KDE.pdf [Accessed 6 June 2020].

[3] En.wikipedia.org. 2020. Kernel (Statistics). [online] Available at: https://en.wikipedia.org/wiki/Kernel_(statistics) [Accessed 6 June 2020].

[4] En.wikipedia.org. 2020. Kernel Density Estimation. [online] Available at: https://en.wikipedia.org/wiki/Kernel_density_estimation [Accessed 6 June 2020].

[5] Finance.yahoo.com. 2020. Yahoo Is Now A Part Of Verizon Media. [online] Available at: https://finance.yahoo.com/ [Accessed 6 June 2020].