Inteligência Artificial (1001336)

Prof. Dr. Murilo Naldi

SVR para Predição de Ações do Facebook

Guilherme Locca Salomão (758569)
João Victor Mendes Freire (758943)
Luís Felipe Corrêa Ortolan (759375)

Apresentação do Problema

- Mercado de ações
- Variação nos preços diariamente
- Aprendizado de Máquina

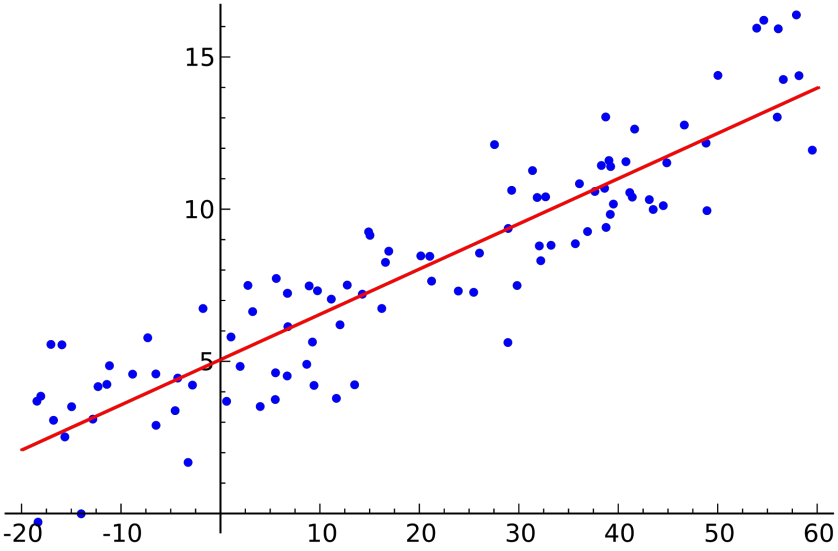
Explicação Teórica

- Plano Cartesiano

- Support Vector Regression (SVR)
  - Regressão Linear
  - Support Vector Machine (SVM)

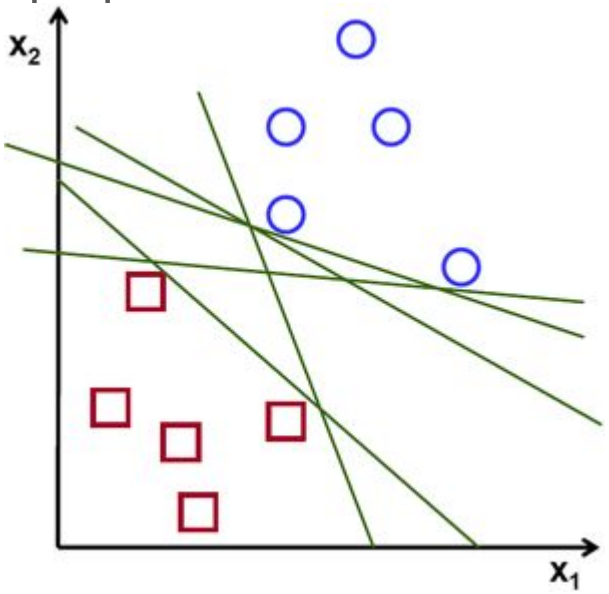
Explicação Teórica - Regressão Linear

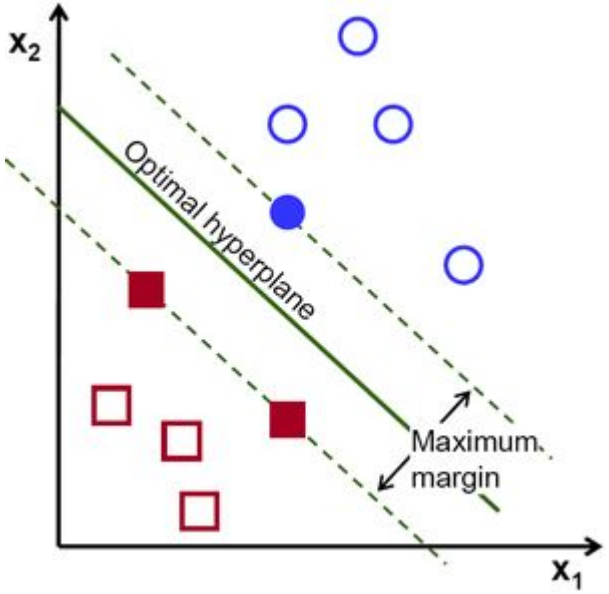
- Técnica estatística para encontrar a reta que melhor representa um conjunto de pontos.



Explicação Teórica - SVM

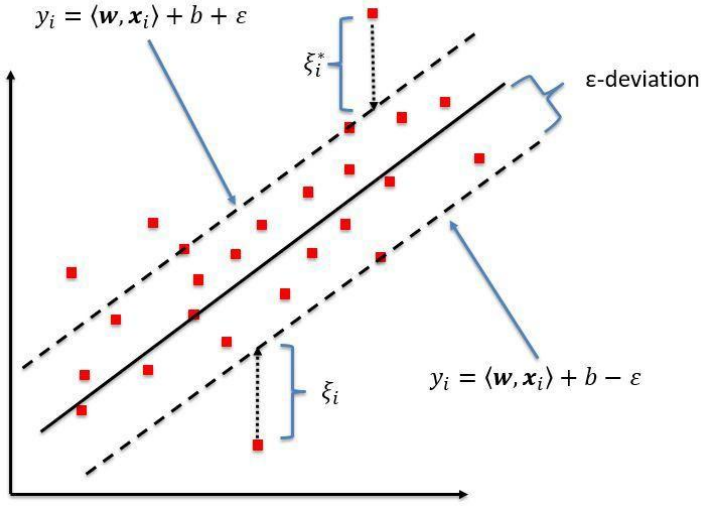
- Técnica de classificação que busca maximizar a distância entre vetores e hiperplano.





Explicação Teórica - SVR

- Combinação de ambas as técnicas para realizar a predição de um valor contínuo. Nesta técnica, o hiperplano vai servir não para classificar, mas para aproximar o próximo valor.



Explicação Teórica - Kernel

Linear

$$f(x) = a \times x + b$$

Polinomial

$$f(x) = x^n + x^{n-1} ... + x + a$$

Base Radial

$$f(x) = e^{(||x-x'||^2 * Y)}$$

Análise dos Resultados

Table 1. Valores das ações do Facebook.

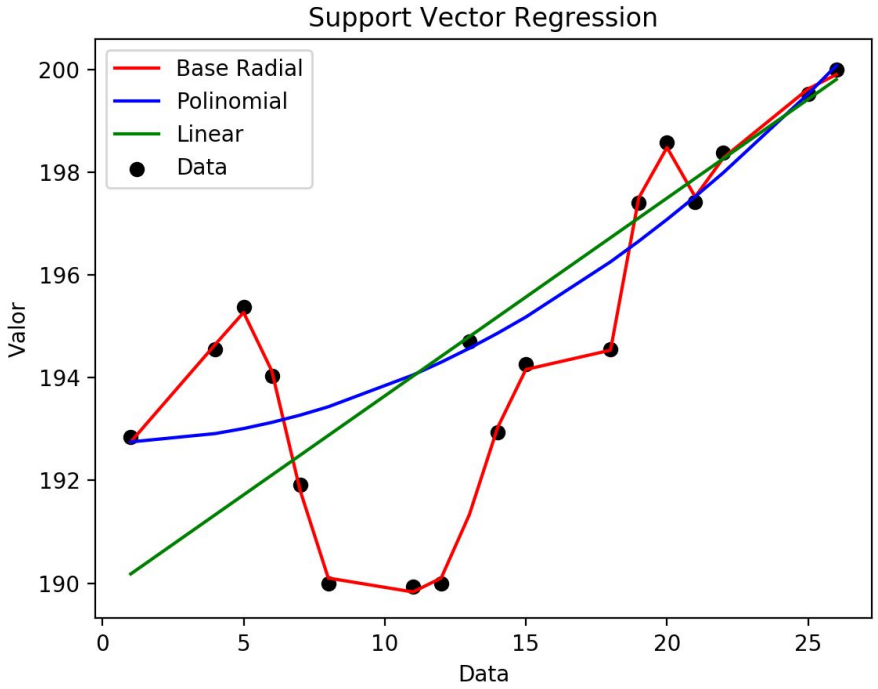
Dia	Valor da Ação	Dia	Valor da Ação
01/11/2019	192,85	14/11/2019	192,92
04/11/2019	194,55	15/11/2019	194,25
05/11/2019	195,36	18/11/2019	194,55
06/11/2019	194,02	19/11/2019	197,39
07/11/2019	191,91	20/11/2019	198,58
08/11/2019	190,00	21/11/2019	197,41
11/11/2019	189,92	22/11/2019	198,38
12/11/2019	190,00	25/11/2019	199,52
13/11/2019	194,69	26/11/2019	200,00

Análise dos Resultados

Table 2. Previsões do algoritmo para as ações do Facebook.

Dia	V. Linear	V. Poli.	V. B. Rad	Valor Real
27/11/2019	202,58	200,62	200,19	199,90
29/11/2019	203,69	201,85	200,96	201,60

Análise dos Resultados



Código - Conjunto de Dados

Date	Open	High	Low	Close	Adj Close	Volume
2019-11-01	192.850006	194.110001	189.910004	193.619995	193.619995	21711800
2019-11-04	194.550003	197.369995	193.809998	194.720001	194.720001	16371300
2019-11-05	195.369995	195.750000	193.600006	194.320007	194.320007	9942000
2019-11-06	194.029999	194.369995	191.350006	191.550003	191.550003	10973000
2019-11-07	191.910004	193.440002	189.470001	190.419998	190.419998	13473000
2019-11-08	190.000000	192.339996	189.699997	190.839996	190.839996	10760800
2019-11-11	189.929993	190.080002	188.539993	189.610001	189.610001	8631200
2019-11-12	190.000000	195.059998	189.740005	194.470001	194.470001	17615500
2019-11-13	194.699997	195.699997	192.740005	193.190002	193.190002	10860700
2019-11-14	192.929993	194.029999	191.449997	193.149994	193.149994	9040500
2019-11-15	194.259995	195.300003	193.380005	195.100006	195.100006	11524300
2019-11-18	194.559998	198.630005	193.050003	197.399994	197.399994	16167200
2019-11-19	197.399994	200.000000	196.860001	199.320007	199.320007	19056800
2019-11-20	198.580002	199.589996	195.429993	197.509995	197.509995	12355400
2019-11-21	197.419998	199.089996	196.860001	197.929993	197.929993	12131000
2019-11-22	198.380005	199.300003	197.619995	198.820007	198.820007	9959800
2019-11-25	199.520004	200.970001	199.250000	199.789993	199.789993	15272300
2019-11-26	200.000000	200.149994	198.039993	198.970001	198.970001	11735500
2019-11-27	199.899994	203.139999	199.419998	202.000000	202.000000	12736600

Código - Construindo Plano Cartesiano

```
df = pd.read_csv('FB.csv')


# Criando as listas / Dataset X e Y
dates  = []
prices = []


# Pega todas as informações, com excessão da última linha.
df = df.head(len(df)-1)


df_dates = df.loc[:,'Date']
df_open  = df.loc[:,'Open']


# Cria o dataset independente 'X' como dates.
for date in df_dates:
 dates.append( [int(date.split('-')[2])] )

# Cria o dataset dependente 'Y' como prices.
for open_price in df_open:
 prices.append(float(open_price))
```

Código - Treinamento

```
# Função para fazer predições usando 3 diferentes modelos de SVR com três kernels diferentes.
def predict_prices(dates, prices, x):
 # Cria 3 modelos SVR
 svr_lin = SVR(kernel='linear', C=1e3)
 svr_pol = SVR(kernel='poly', C=1e3, degree=2)
 svr_rbf = SVR(kernel='rbf', C=1e3, gamma=0.1)

 # Treina os modelos com dates e prices.
 svr_lin.fit(dates,prices)
 svr_pol.fit(dates,prices)
 svr_rbf.fit(dates, prices)
 ...
```

Código - Predição

```
def predict_prices(dates, prices, x):
 ...
 plt.scatter(dates, prices, color = 'black', label='Data')
 plt.plot(dates, svr_rbf.predict(dates), color = 'red', label='Base Radial')
 plt.plot(dates, svr_pol.predict(dates), color = 'blue', label = 'Polinomial')
 plt.plot(dates, svr_lin.predict(dates), color = 'green', label='Linear')
 plt.xlabel('Data')
 plt.ylabel('Valor')
 plt.title('Support Vector Regression')
 plt.legend()
 plt.show()

 #Retorne os três modelos de predição
 return svr_rbf.predict(x)[0], svr_pol.predict(x)[0], svr_lin.predict(x)[0]


print(predict_prices(dates, prices, [[29]]))
```

Bibliografia

- ITNEXT Stock Prediction Using Python.
  Disponível em: (https://itnext.io/facebook-stock-prediction-bcfc676bc611).
- Towards Data Science.
  Disponível em:
  (https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47).
- Awad M., Khanna R. (2015) Support Vector Regression. In: Efficient Learning Machines. Apress, Berkeley, CA.
  Disponível em: (https://link.springer.com/chapter/10.1007/978-1-4302-5990-9_4).
- Wikipedia: Linear Regression (imagem).
  Disponível em: (https://en.wikipedia.org/wiki/Linear_regression).