Capstone Project

Machine Learning Engineer Nanodegree

Ricardo Paschoeto

September 10, 2018

Preditor de Preços de Ações

I. Definição

Visão Geral

Este projeto veio do meu interesse no mercado financeiro e suas características técnicas ligadas aos movimentos de ativos (ações). Eu desenvolvi um aplicativo que irá prever os preços das ações em um intervalo de datas ou em uma data específica escolhida para ações do mercado Brasileiro. Os dados usados para análise vêm de dados históricos obtidos pelo sistema tendo como fonte provedores específicos para coleta de informações via Phyton (por exemplo, yahoo, Google finance, entre outros) para cada ação. O usuário interage com o sistema por meio de um aplicativo web. A figura 1 ilustra os dados históricos de preços para algumas ações negociadas na BOVESPA.

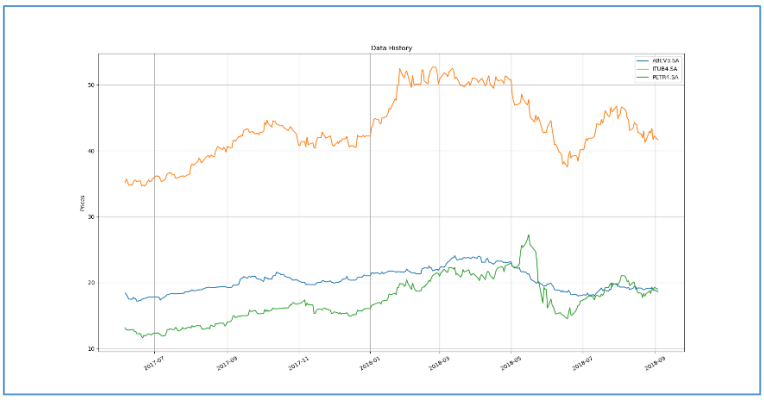


Figura 1 - Histórico de Preços

Descrição do Problema

O objetivo é prever os preços de fechamento das ações tendo como entradas - Preço de Abertura, Maior Preço, Menor Preço, Preço de Fechamento, Volume, Volatilidade, Preço do Minério de Ferro, Taxa de Câmbio e Preço do Barril de Petróleo – executado no Aplicativo Web. A seqüência do processo são as seguintes:

1. Treinamento e teste da interface do usuário:

a. O usuário irá selecionar um intervalo de datas para consultar ou escolher coletar os dados históricos previamente gravados no sistema
b. Escolha do comprimento do Teste – 5, 10, 14, 28, 30, 60 e 90 dias
c. Uma tabela será apresentada com o resumo dos resultados - RMSE e R^2 score - para cada modelo disponível no sistema - Regressão Linear (Benchmark), Modelo de Árvore de Regressão, Modelo de Árvore de Regressão AdaBoost, SDG, SDG AdaBoost
d. Ir para a página de resultados, onde é possível escolher o modelo, o período para prever os preços ou a data específica de previsão, os resultados aparecerão na forma gráfica

O aplicativo ajudará o usuário a tomar uma decisão sobre o movimento de preços das ações e assim poderá decidir comprar, vender ou manter as ações.

Métricas

Uma das execuções básicas do sistema envolverá uma ou mais chamadas para a interface de pesquisa e Treinamento. Dentro dessas interfaces, o usuário pode verificar a precisão da previsão dentro do intervalo de datas do Teste, após as datas de treinamento, por exemplo, 5, 7, 10, 14, 28, 30, 60, 90 dias. Os resultados serão apresentados em forma de tabela com colunas - RMSE e pontuação R2- para cada linha que contém a ação.

Root Mean Square Error (RMSE) é o desvio padrão dos resíduos (erros de previsão). Residuais são uma medida de quão longe estão os pontos de dados da linha de regressão; RMSE é uma medida de como se espalham esses resíduos. Em outras palavras, ele informa a concentração dos dados na linha de melhor ajuste. O RMSE é comumente usado em climatologia, previsão e análise de regressão para verificar os resultados experimentais.A fórmula é:

$$\mathrm{RMSE} = \sqrt{(\mathrm{p} - \mathrm{o})}$$

Onde:

- f = previsões (valores esperados ou resultados desconhecidos),
- o = valores observados (resultados conhecidos).

R2 é uma medida estatística de quão próximos os dados estão da linha de regressão ajustada. É também conhecido como o coeficiente de determinação, ou o coeficiente de determinação múltipla para regressão múltipla. A definição de R2 é bastante direta, é a porcentagem da variação da variável resposta que é explicada por um modelo linear. Ou:

$$\mathrm{R2} = \text{Variação explicada / Variação Total}$$

R2 está sempre entre 0 e 100%:

- 0% indica que o modelo não explica nada da variabilidade dos dados de resposta em torno de sua média.
- 100% indica que o modelo explica toda a variabilidade dos dados de resposta em torno de sua média.

Em geral, quanto maior o R2, melhor o modelo se ajusta aos seus dados

II. Analysis

Exploração dos Dados

O conjunto de dados provêm da internet e podem ser acessados pelo Sistema na forma de um conjunto de arquivos gerados com esses dados e armazenados em formato ".csv". Novas consultas na rede podem ser realizadas ou os dados já marzenados poderão ser utilizados dentro do sistema. Conjuntos históricos com valores incompletos, como feriados e fins de semana, serão tratados internamente dentro do sistema. Abaixo há o detalahmento de cada dado e sua importância para o sistema:

- **Preço de Abertura**

O preço de abertura de um ativo corresponde ao primeiro negócio realizado no dia.

Este valor é um número no formato *float*.

- Maior Preço

O Maior preço é o valor mais alto negociado atingido por um ativo no final do pregão no período de análise (minutos,horas,dias,semanas,meses, etc).

Este valor é um número no formato *float*.

- Menor Preço

O menor preço se mostra contrário ao Maior Preço, sendo assim o Menor Preço é o menor valor negociado atingido por um ativo no período de análise. As mesmas considerações feitas para o Maior Preço se encaixam para o Menor Preço.

Este valor é um número no formato *float*.

- Preço de Fechamento Ajustado

O Preço de Fechamento Ajustado é o último preço negociado de um ativo no período de análise e é ajustado pois caso haja qualquer tipo de bonficação ou dividendos pagos pela empresa ,este valor será descontado do preço de cada ativo. Este é valor que buscamos realizar a previsão.

Este valor é um número no formato *float*.

Figura 2 - Exemplo de Gráfico Candlestick

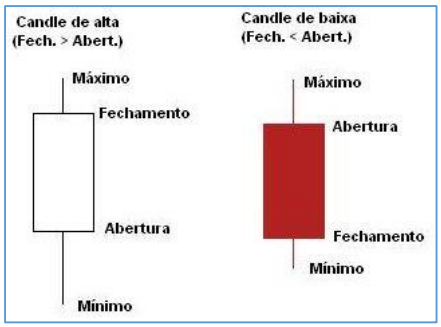


Figura 3 - Preços Máximo, Mínimo, Abertura e Fechamento em Candles

As figuras acima mostram um gráfico em candlestick usado para verificar os valores de Máximo, Mínimo, Abertura e Fechamento de preços ao mesmo tempo.

- Volume

O volume é um indicador gráfico que apresenta o número de ações negociadas em um período de análise especificado (minutos,horas,dias,semanas,meses, etc) para um ativo em particular. Este valor é um número no formato *float*, porém ele apresenta valores da ordem de 10^6.

- Volatilidade

Volatilidade é um termo geral usado para descrever oscilações diárias nos preços (independentemente da direção). Para a aplicação será usada a volatilidade calculada pelo modelo de GARCH(1,1), este modelo atribui pesos maiores aos dados mais recentes no cálculo da volatilidade histórica.

Equação GARCH(1,1):

$$\sigma_t^2 = \alpha_0 + \alpha_1 \alpha_{t-1} + \beta_1 \sigma_{t-1}^2 \text{ (01)}$$

onde $\alpha_0 > 0$, $\alpha_1 > 0$, $\beta_1 > 0$ e $\alpha_1 + \beta_1 < 1$

Este valor é um número no formato *float*, na forma de Porcentagem.

- Preço do Minério de Ferro

Como matéria-prima básica para empresas do ramo siderúrgico seu preço tem impacto muito forte nos preços das ações. O Preço do minério de ferro comercializado na China por exemplo, influencia nos preços de grandes mineradoras no mundo.

Este valor é um número no formato *float*.

- Taxa de Câmbio

Variação na taxa de câmbio

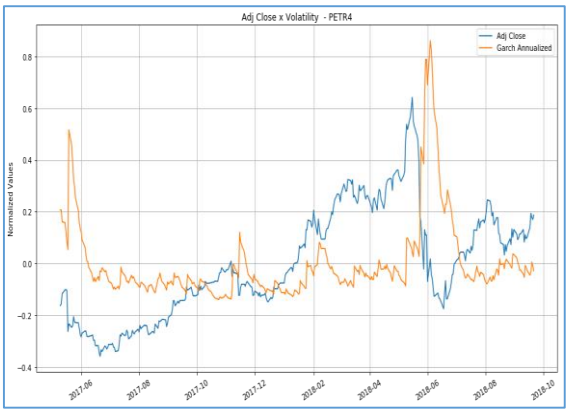
Este valor é um número no formato *float*.

- Preço do Barril de Petróleo

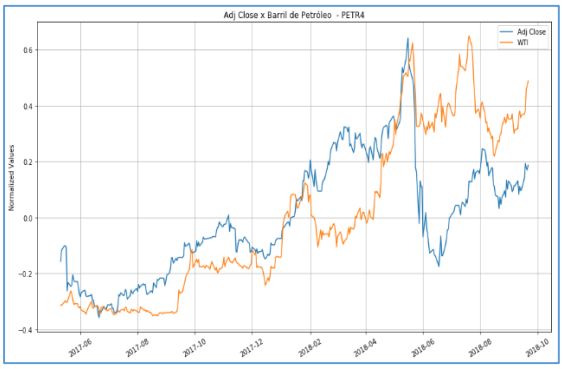
Para empresas ligadas ao setor petrolífero, o preço do barril de petróleo é fortemente correlacionado com seus ativos. Este valor é um número no formato *float*.

Visualização Exploratória

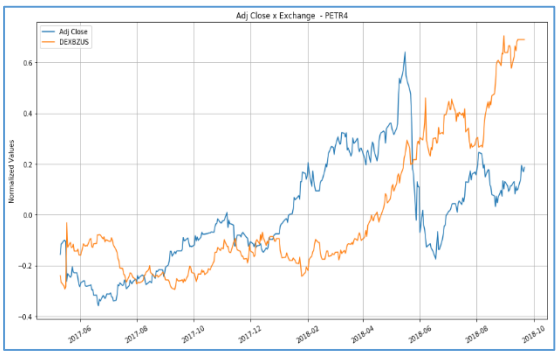
Como exemplo dos dados de entrada e saída e sua relação, abaixo temos a representação dentro de um intervalo de tempo dos dados de Preço de Fechamento Ajustado (Adj Close) e da Volatilidade (Garch Annualized) e do Preço do Barril de Petróleo para Petrobrás, por exemplo.



Através deste primero exemplo podemos perceber o comportamento da Preço de Fechamento Ajustado relacionado com a Volatilidade, no período em que a volatilidade é baixa – entre junho de 2017 e maio de 2018 – ocorre um aumento gradativo do preço da PETR4 até que junho de 2018 ocorre uma queda abrupta acompanhada de um pico no valor da volatilidade. Esse fato caracteriza um evento político (mudança de presidente e equipe econômica) no país ou algum evento corporativo muito negativo dentro da empresa (demissão de executivo de alto escalão ou mudança na gestão de dívidas) que aumenta o risco (Volatilidade) para o investidor de manter as ações em sua carteira de investimento.



Um segundo exemplo é mostrado com o Preço de Fechamento ajustado e o Preço do Barril de Petróleo, podemos ver que a longo prazo a tendência da PETR4 é acompanhar a oscilação do preço do Barril de Petróleo, isto não ocorre em períodos onde ocorrem eventos externos, como citados anteriormente , isto pode ser visto em junho de 2018. No entanto passado esse evento o Preço tende a acompanhar o Preço do Barril de Petróleo novamente.



O terceiro exemplo mostra o Preço de fechamento ajustado pelo Câmbio, novamente podemos notar a longo prazo que uma valorização na Ação segue o aumento da relação Real/Dólar, principalmente pelo fato de que o produto produzido pela petrobrás – o Petróleo – segue como vimos acima o preço do Barril de petróleo e este é cotado em dólar.

Algoritmos e Técnicas

Este é um problema contínuo, ou seja, os valores evoluem de forma suave e podem atingir infinitos valores dentro de um intervalo. O *GridSearchCV* será aplicado a todos os modelo de forma a ajudar no *tunning/otimização* dos parâmetros dos mesmos. Para este caso os algoritmos usados serão os aplicados a este tipo de problema, como:

- **Linear Regression** – A Regressão linear é uma ótima escolha para problemas com entradas lineares, pois é um modelo muito simples. É muito boa para trabalhar e explorar os dados sem se preocupar com os detalhes maiores do modelo.
  A desvantagem é justamente o fato de simplificar muito problemas reais. Ela parte do pressuposto que existe uma relação direta entre as variáveis. As vezes isto não está correto. Para o modelo deste trabalho a relação linear entre as variáveis, como já visto, é muito forte, sendo assim a Regressão Linear poderá ser usada sem maiores problemas.
- **Decision Tree Regressor** – Um dos aspectos mais úteis das árvores de decisão, Figura 4, é que eles forçam você a considerar o maior número possível de resultados possíveis de uma decisão. Uma árvore de decisão pode ajudá-lo a avaliar as consequências prováveis de uma decisão em relação a outra. Em alguns casos, pode até ajudá-lo a estimar retornos esperados de decisões. Fornecem uma estrutura para considerar a probabilidade e os resultados das decisões.
  Uma desvantagem é que como os resultados decisões e decisões subsequentes são baseados na probabilidade/expectativa estes resultados podem ser diferente das decisões reais.
  Outro ponto é que as árvores de decisão são relativamente fáceis de entender quando há poucas decisões e resultados incluídos na árvore. Grandes árvores que incluem dezenas de nós de decisão (pontos onde novas decisões são tomadas) podem ser complicadas e podem ter valor limitado. Quanto mais decisões houver em uma árvore, menos precisos serão os resultados esperados.

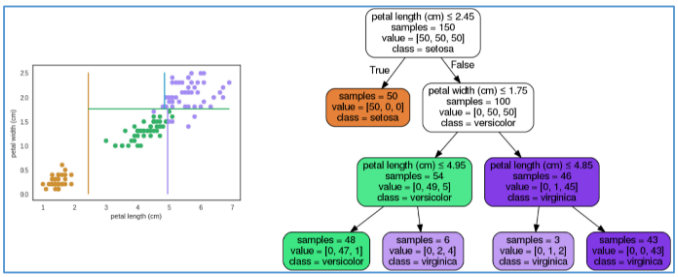


Figura 4 - Decion Tree

- **Decision Tree Regressor com AdaBoost** – Algoritimo *Adaboost.* AdaBoost, abreviação de Adaptive Boosting ele pode ser usado em conjunto com muitos outros tipos de algoritmos de aprendizado para melhorar o desempenho. A saída dos outros algoritmos de aprendizado (Week Learners') é combinada em uma soma ponderada que representa a saída final do classificador. [1]Quando usado com o aprendizado da árvore de decisão (Week Learner), o crescimento da árvore é realizado de modo que as subárvores posteriores tendem a se concentrar em exemplos mais difíceis de classificar.
- **Modelo SGD (Stochastic Gradient Descent)** – O Gradient Descent é usado durante o treinamento de um modelo de aprendizado de máquina. É um algoritmo de otimização, baseado em uma função convexa, que ajusta os parâmetros de forma iterativa para

minimizar uma determinada função ao mínimo local. Explicando de forma mais simples:[2] Imagine um homem que queira descer uma colina, com o menor número de passos possíveis. Ele apenas começa a descer a colina dando grandes passos na direção mais íngreme. Ao chegar a base, ele fará etapas cada vez menores. Esse processo pode ser descrito matematicamente, usando o gradiente. A Figura 5 ilustra o processo de forma simplificada.

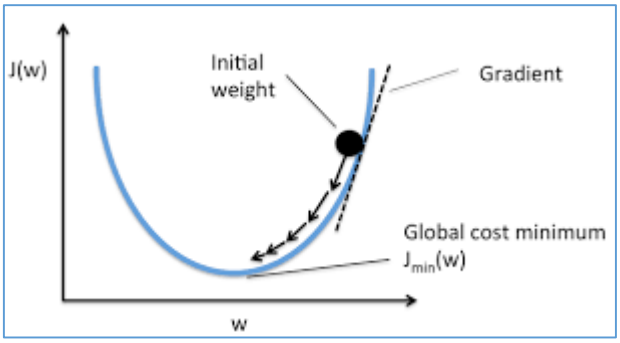
- 

Figura 5 - Ilustarção SGD

- **Modelo SGD com AdaBoost** – Algoritimo *Adaboost*. Aplica-se a partir dos fundamentos citados acima. Com o SGD com um "*Week Learner*".

Cada algoritmo irá receber de forma muito semelhante os dados. Os dados de terinamento serão determinados pelo usuário através da interface do programa e através deste valor será calculado o tamanho dos dados de treinamento associado. Como entrada também recebem o dataframe com os dados históricos do(s) ativo(s). Para os algoritmos que utilizam o SGD será realizado um pré - tratamento *StandardScaler* para ajustar os dados ao algoritmo. Assim no fim, cada algoritmo irá trabalhar com dados separados em Teste e Treinamento.

O sistema é formado por classes que representam cada uma um conjunto de características do sistema e que serão usadas pelas classes do Django de controle e apresentação. A classe *data_process* coleta os dados e realiza o tratamento dos mesmos, limpeza por exemplo, a classe *data_statistics* calcula a volatilidade, a classe *supervised_models* contém os modelos usados no sistema descritos acima, a classe *views* do Django faz o link entre as classes citadas e a camada de apresentação ao usuário, fechando 0 modelo *MVC (Model View e Controller).*

Benchmark

O Benchmark utilizado será o algoritmo *LinearRegression* , este é um dos modelos mais simples para problemas do tipo contínuo. O Benchmark poderá ser comparada com os outros modelos, de forma que os preços previstos fiquem a uma distância do preço real considerada satisfatória pelo usuário.

III.Metodologia

Pré-Processamento dos dados

Em relação aos históricos de dados coletados do mercado financeiro como mencionado anteriormente podem ocorrer *gaps* dentro do conjunto de dados, sendo assim, na função de coleta e escrita dos dados em arquivos *.csv* é aplicada a função *fillna* com os métodos internos 'ffill' e 'bfill' . A primeira propaga o último valor válido para frente até o próximo valor válido. A segunda utiliza o próximo valor válido para preencher os valores passados, figura 6 e 7. Outro processamento foi o cálculo da volatilidade histórica pelo modelo de GARCH(1,1) descrito anteriormente.

Para os algoritmos de *Stochastic Gradient Descent* (SGD) que são sensíveis ao dimensionamento, foi aplicado o método *StandardScaler.*

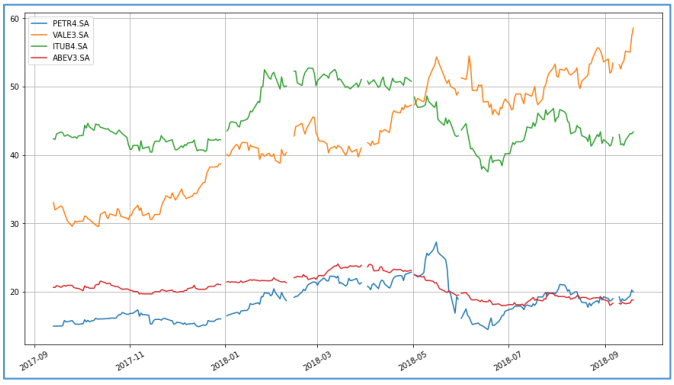


Figura 6 - Dados com Gaps

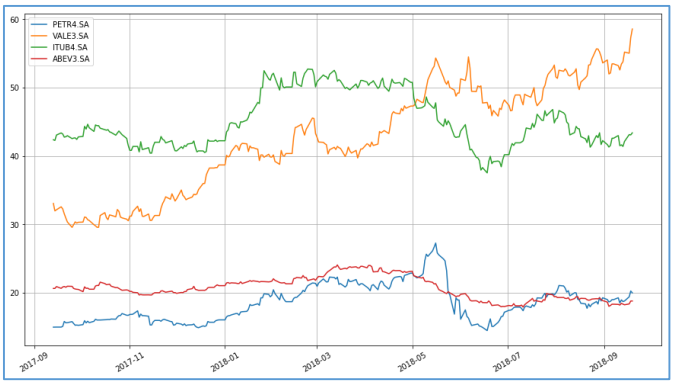


Figura 7 - Após processamento com 'fillna'

Implementação

O processo de implementação foi dividido em três etapas:

- Apresentação dos dados coletados em forma de gráfico histórico das respectivas ações e uma tabela comos dados dos útimos valores para cada ação. Desta forma apresenta-se uma visão geral dos dados coletados.

- Interface para treinamento , onde o tamanho do conjunto de teste é escolhido e assim para o conjunto de dados cada algoritmo disponível no sistema será aplicado e os resultados – *RMSE e R^2score* – serão apresentados para comparação com o Benchmark e escolha do modelo a ser aplicado.
- Interface para aplicação do Modelo onde o algoritmo escolhido será ser aplicado e o resultado será apresentado gráficamente com a os valores previstos e os reais plotados juntos dentro do período de teste escolhido.

A arquitetura de software escolhida para a aplicação foi o modelo *MVC (Model – View - Controller)* muito utilizado em sistema de aplicativos Web. Todo o sistema foi desenvolvido dentro da IDE *Vscode* da Microsoft. O *Django* foi utilizado para a construção dos *Templates (View)* e das *Views (Controller)* que integram o código *Python (Model)* do sistema. A arquitetura básica simplificada é mostrada abaixo.

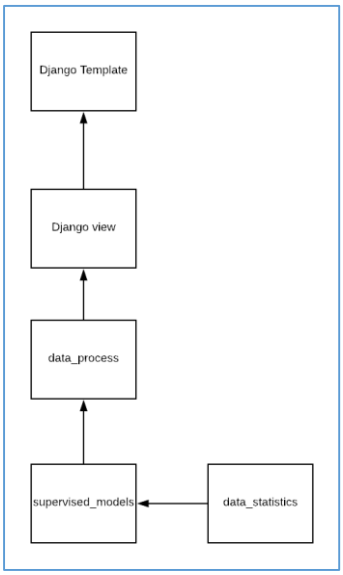


Figura 8 - Modelo UML

A Classe *data_statistcs* contém funções relacionadas com resultados oriundos da estatística, como volatilidade , retorno diário da ação e média móvel. Esta é uma classe auxiliar.

A Classe *supervised_models* contém a implementação dos modelos usados no sistema:

- Benchmark (Regressão Linear)
- tree_model (Modelo de Árvore de Regressão)
- tree_model_adaboost (Modelo de Árvore de Regressão com Adaboost)
- sgd_model (Modelo de Gradiente de Descida)
- sgd_model_adaboost (Modelo de Gradiente de Descida com Adaboost)

A Classe *data_process* contém a função de consulta e pré-processamento e outras funções auxiliares e utiliza objetos das Classes *data_statistcs* e *supervised_models*. Ela faz o encapsulamento dessas funções para uso dentro da Classe *Views* do *Django*.

A Classe *View* será a ponte entre a classe *data_process* e o *Template*. O *Template* conterá as interfaces explicadas acima, é basicamente composto de código HTML com variáveis internas

para link com a Classe *View*. O processo de treinamento e teste será realizado pelo próprio usuário antes de seguir para a apresentação dos resultados de forma comparativa.

Todo o processo de criação das funções foi realizado utilizando um *jupyter Notebook* - ***stocks_preditor.py*** - de forma a obter maior flexibilidade para os testes e depuração dos erros com o código *Python*.

Refinamento

A primeira solução foi realizada utilizando somente o modelo de Regressão Linear, pois este é o modelo utilizado como Benchmark. Nesta primeira etapa os dados de teste e treinamento foram separados aleatóriamente pela função *train_test_split*. Porém neste processo não se tem definido uma sequência cronológica tanto dos dados de treinamento, quanto de teste. Os dados de teste devem estar históricamente depois dos dados de treinameto. Sendo assim os dados de treinamento e teste foram separados em função do comprimento do teste escolhido, sendo os dados de teste a sequência final dos dados com este comprimento.

IV.Resultados

Avaliação e Validação dos Modelos

O sistema apresenta como resultados as métricas dos vários modelos internos. Caberá ao usuário determinar qual modelo atende melhor as expectativas se apoiando nos resultados de avaliação fornecidos. Os resultados variam de ação para ação. Como apresentado abaixo temos um exemplo de uma tabela de validação gerada pelo sistema , para um tamanho de teste de 60 dias , com 263 dados históricos coletados:

Tabela 1- Dados de Avaliação e Validação

Stocks	Benchmarch		tree		tree Adaboost		SGD		SGD Adaboost	
	RMSE	R2 score	RMSE	R2 score	RMSE	R2 score	RMSE	R2 score	RMSE	R2 score
ABEV3.SA	0.0951	0.9218	0.0521	0.9766	0.0538	0.9749	0.1745	0.9889	0.1653	0.9901
ITUB4.SA	0.2311	0.9783	0.2193	0.9805	0.2019	0.9835	0.6163	0.9723	0.5972	0.974
PETR4.SA	0.2058	0.9826	0.107	0.9953	0.1176	0.9943	0.3068	0.9857	0.2857	0.9876
VALE3.SA	0.3545	0.9882	0.2819	0.9926	0.2538	0.994	0.8057	0.9876	0.7282	0.9899

Tabela 2- Melhores Estimadores pelo GridSearchCV

Modelo	***Melhor Estimador - GridSeacrhCV***
Linear Regression	LinearRegression(copy_X=True, fit_intercept=False, n_jobs=1, normalize=True)
DecisionTreeRegressor	DecisionTreeRegressor(criterion='mse', max_depth=3, max_features=None,
	max_leaf_nodes=None, min_impurity_decrease=0.0,
	min_impurity_split=None, min_samples_leaf=1,
	min_samples_split=3, min_weight_fraction_leaf=0.0,
	presort=False, random_state=None, splitter='best')
SGD Regressor	SGDRegressor(alpha=0.0001, average=False, epsilon=0.1, eta0=0.01,
	fit_intercept=True, l1_ratio=0.15, learning_rate='invscaling',
	loss='squared_loss', max_iter=1490, n_iter=None, penalty='l2',
	power_t=0.25, random_state=None, shuffle=True, tol=None, verbose=0,
	warm_start=False)

Para o modelo de *Benchmark* e *Árvore de Decisão* foram implementados utilizando um modelo de validação *Grid Search* para evitar um possível *overfitting* – A Tabela 2, apresenta o resultado dos estimadores otimizados selecionados pelo algoritimo GridSearchCV para cada modelo do sistema.A variação dos comprimentos de teste e do próprio conjunto de dados é realizada dinamicamente no sistema e apresenta resultados de avaliação, apresentando abaixo na Figura 9 o R^2 score e sumarizando na Tabela 3 o R^2 score e o RMSE:

Tabela 3 - Sumário

PETR4	5		7		14		28		30		31		90	
	RMSE	R2 score	RMSE	R2 score	RMSE	R2 score	RMSE	R2 score	RMSE	R2 score	RMSE	R2 score	RMSE	R2 score
Regressão Linear	0,4103	0,9449	0,3574	0,9701	0,3076	0,9775	0,2710	0,9820	0,2334	0,9855	0,2311	0,9787	0,2962	0,9811
Árvore de Decisão	0,9227	0,7215	0,7504	0,8684	0,8124	0,8433	0,7047	0,8785	0,4308	0,9506	0,4363	0,9243	0,4926	0,9477
Árvore de Decisão AdaBoost	0,5113	0,9145	0,4346	0,9559	0,4572	0,9504	0,4761	0,9445	0,3673	0,9641	0,3700	0,9455	0,3448	0,9744
SGD	0,6490	0,8622	0,5934	0,9177	0,6107	0,9114	0,6145	0,9076	0,3769	0,9622	0,3702	0,9455	0,4138	0,9631
SGD AdaBoost	0,7605	0,8108	0,6571	0,8991	0,6589	0,8969	0,6341	0,9016	0,4849	0,9374	0,3783	0,9431	0,4943	0,9474

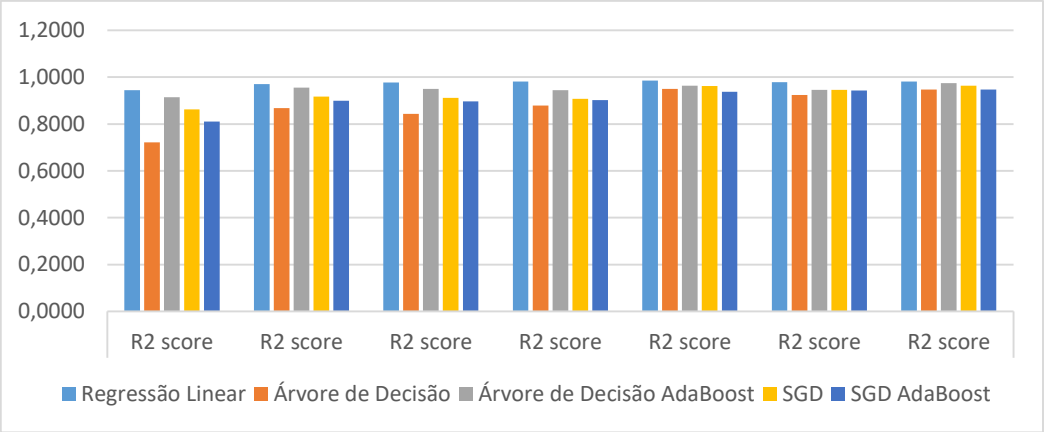


Figura 9

Pelo gráfico o modelo de *Regressão Linear* , mesmo sendo o modelo mais simples, apresenta o melhor resultado, em média, dentre os outros modelos do sistema. A seguir os resultados apresentados na página *Predict* onde são plotados juntos os conjuntos de testes e os dados de predição para comparação dentro do range de datas de teste de 30 dias, por exemplo.

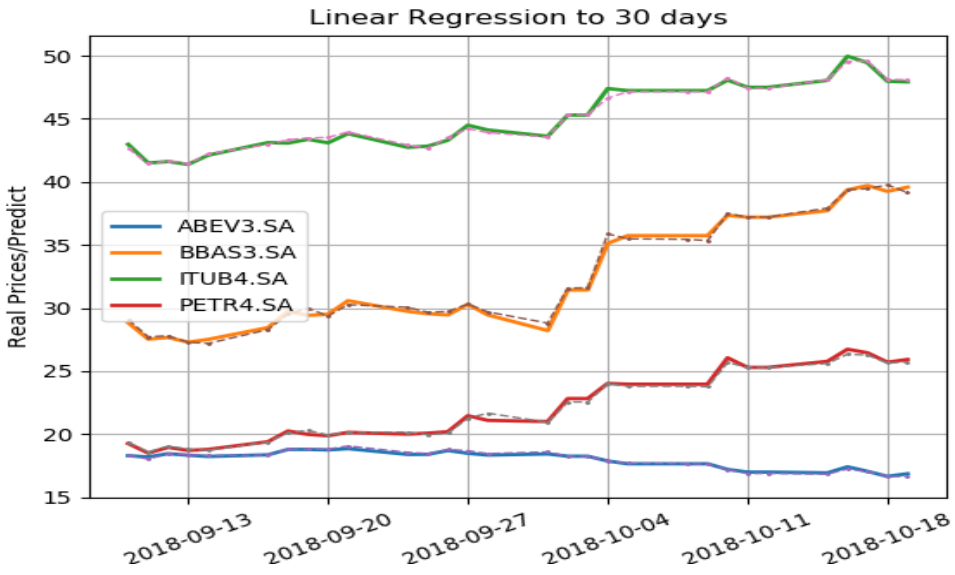


Figura 10 - Linear Regression

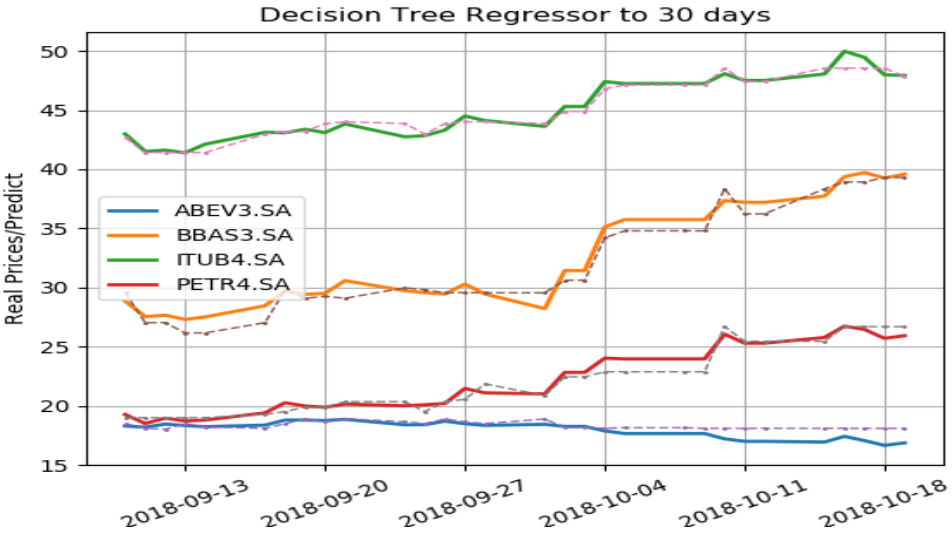


Figura 11 - Decision Tree

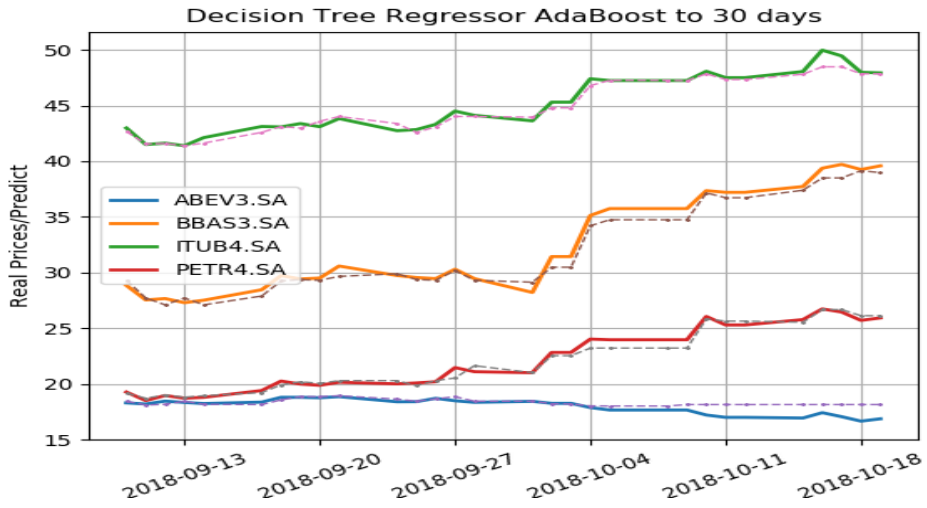


Figura 12 - Decision Tree Regressor Adaboost

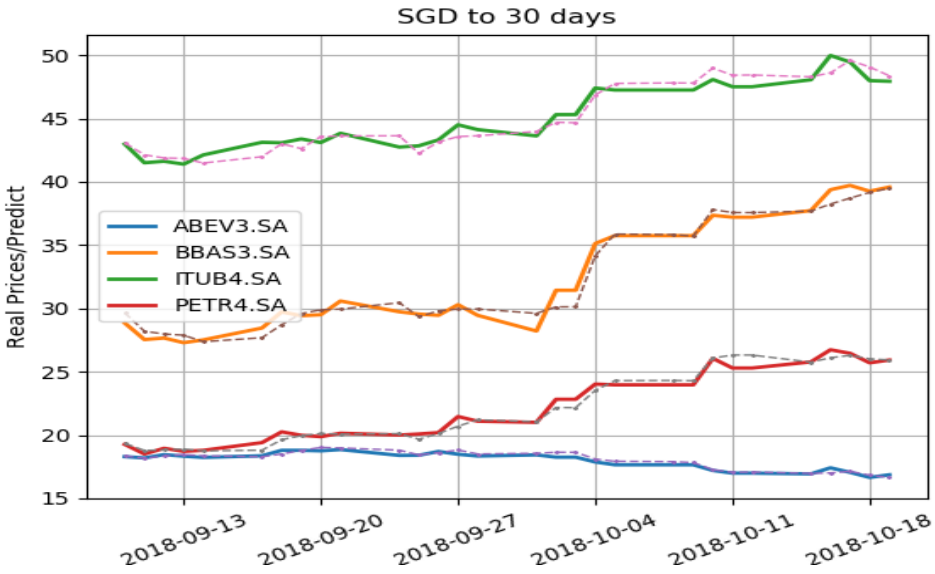


Figura 13 – SGD

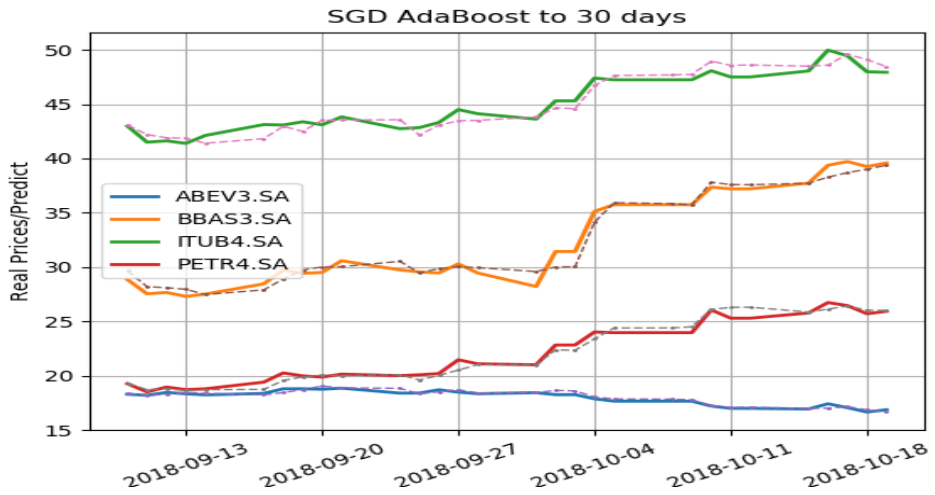


Figura 14 - SGD Adaboost

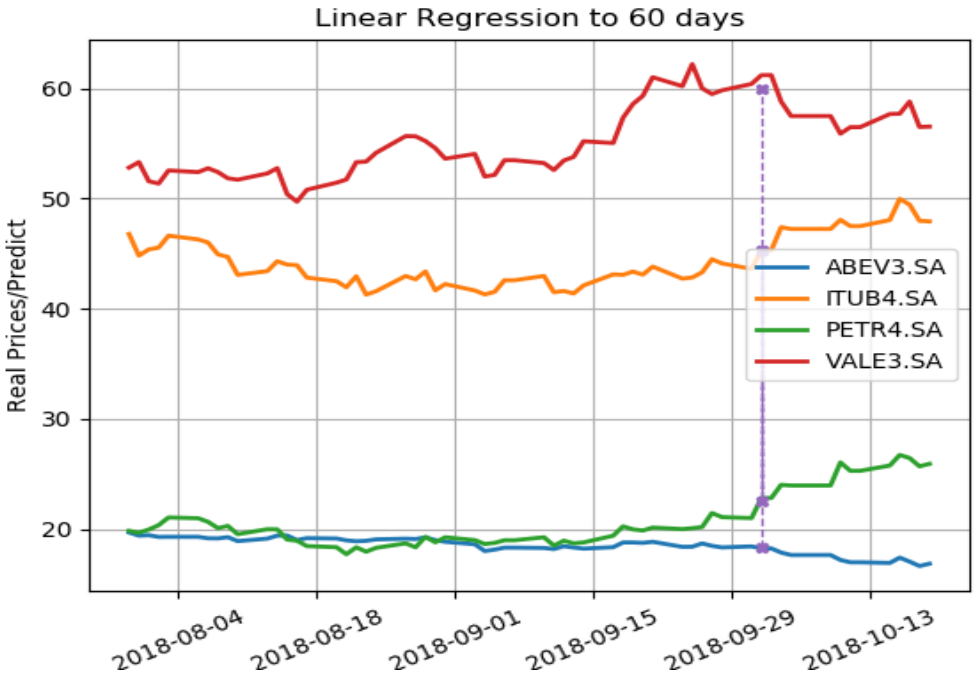


Figura 15 - Previsão para uma data Específica

Stocks and Period

Stocks: Start Date: dd/mm/aaaa End Date: dd/mm/aaaa searching

Read from Directory

Table with last price of the Stocks:

Date	Stock	High	Low	Open	Close	Volume	Adj Close	Iron Ore	Oil Barrel	Currency Exchange	Historical Volatility
2018-10-19	ABEV3.SA	16.96	16.56	16.8	16.86	8716200.0	16.86	17.32	7.74	3.782	21.47
2018-10-19	BBAS3.SA	39.86	39.16	39.83	39.58	9764100.0	39.58	17.32	7.74	3.782	50.177
2018-10-19	ITUB4.SA	48.67	47.74	48.35	47.93	6995600.0	47.93	17.32	7.74	3.782	33.661
2018-10-19	PETR4.SA	26.3	25.72	26.15	25.92	61676900.0	25.92	17.32	7.74	3.782	51.355

Figura 16 - Tela de entrada e Tabela de Cotações

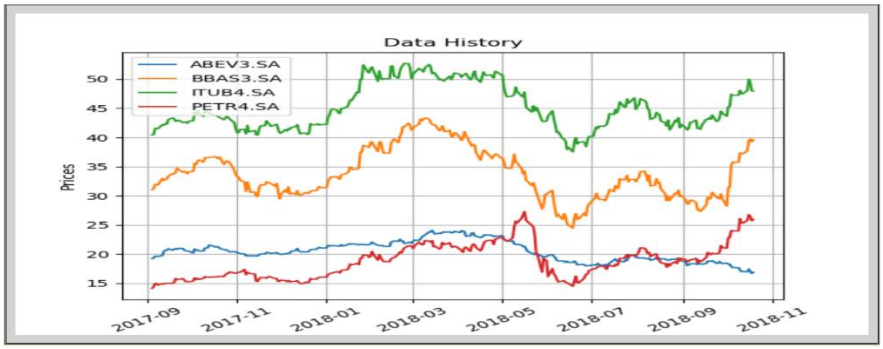


Figura 17 - Cotações históricas

Model Parameters Selection:

Test Size: 5 run analisys

Model Analysis:

Stocks	Benchmarch		tree		tree Adaboost		SGD		SGD Adaboost	
	RMSE	R2 score	RMSE	R2 score	RMSE	R2 score	RMSE	R2 score	RMSE	R2 score
ABEV3.SA	0.1054	0.9747	0.6443	0.0539	0.6164	0.1342	0.2116	0.898	0.2305	0.8789
BBAS3.SA	0.3051	0.9949	0.7702	0.9676	0.6326	0.9781	0.6814	0.9746	0.6442	0.9773
ITUB4.SA	0.227	0.992	0.6913	0.9259	0.4519	0.9683	0.6707	0.9303	0.7217	0.9193
PETR4.SA	0.2006	0.9946	0.4767	0.9696	0.3956	0.9791	0.4402	0.9741	0.424	0.976

go to Predict

Figura 18 - Tabela com as Métricas de Avaliação e Entrada para comprimento dos testes

Os resultados alcançados estão dentro do esperado para cada modelo apresentado. O *GridSearchCV* ajudou a manter os parâmetros dentro dos valores apropriados. Os resultados para variações nos comprimentos dos dados apresentados no início desta seção mostram que o sistema se apresenta robusto e a confiabilidade foi verificada pelo R^2 score para cada modelo e conjuntos de treinamento e teste.

Justificativa

O sistema em sua primeira proposta de desenvolvimento definiria um modelo único dentre vários modelos de Regressão como um modelo final, sendo avaliado por meios estatísticos e comparado ao Benchmark representado pela Regressão Linear. Mas de acordo com os resultados apresentados na Figura 9 o Benchmark se manteve mais estável com a variação do comprimento dos dados de treinamento e teste, isto sendo verificado pelos resultados do R^2 score.

Dessa forma todos os modelos criados para predição dos dados foram mantidos afim de enriquecer o sistema em termos de resultados e como forma de estudo comparativo.

O modelo *Linear Regression*, escolhido como Benchmark, apresenta os melhores resultados, como mencionado acima. Este fato pode ser explicado pela linearidade dos dados, as variáveis como *Preço de Abertura, Fechamento, Máximo e Mínimo* tem forte correlação e apresentarão maior peso nos coeficientes lineares sendo assim mais influentes na resposta. Considerando que estes pesos são ajustados, por exemplo, pelo *Gradiente Estocástico de Descida*. Outras variáveis como *Preço do Barril de petróleo (WTI), Preço do Minério de Ferro (FOE), Dólar (DEXBZUS), Volatilidade (Garch (1,1))* irão apresentar menores correlação e terão menores pesos na equação linear:

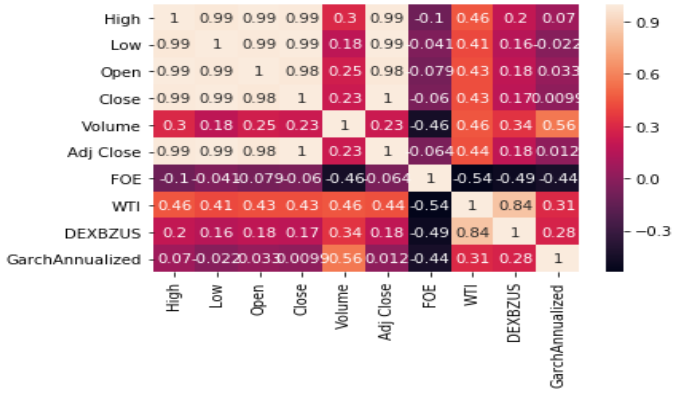


Figura 19 - Correlação Entradas x Saída

Para algoritmos como *Decision Tree Regressor* apresentam uma variação relativamente maior na performance fato que pode ser explicado pela própria limitação do algoritmo quando se variam os dados de treinamento. Uma visível melhora na performance se observa quando a técnica *AdaBoost* é aplicada no modelo, o algoritmo se mostra útil em melhorar a performance principalmente em *week Learners* como as *Decision Tree*. Este foi o modelo que mais se aproximou da performance do modelo *Linear Regression*.

V. Conclusão

Forma Livre de Visualização

O sistema apresenta os resultados em forma de gráficos históricos e tabelas. Os resultados se apresentaram como esperado de acordo com o modelo escolhido e como discutido acima. Uma importante visualização verificada na entrada *Volatilidade* e sua relação com preço ajustado. Pelo sistema se confirma na prática a teoria de que como a *Volatilidade* representa o risco do mercado e influencia inversamente os preços dos ativos – maior volatilidade (risco) menor o preço do ativo – onde altas bruscas na volatilidade fazem os preços caíram vertiginosamente independente das outras variáveis de entrada, como o *preço do Barril de Petróleo e Câmbio* no caso do exemplo da Petrobrás, resultado análogo acontece para quedas bruscas na volatilidade.

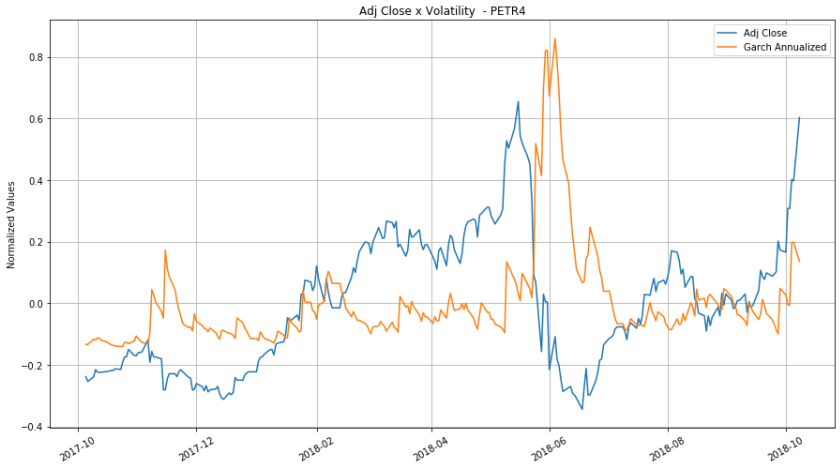


Figura 20 - Preço Adj x Volatilidade

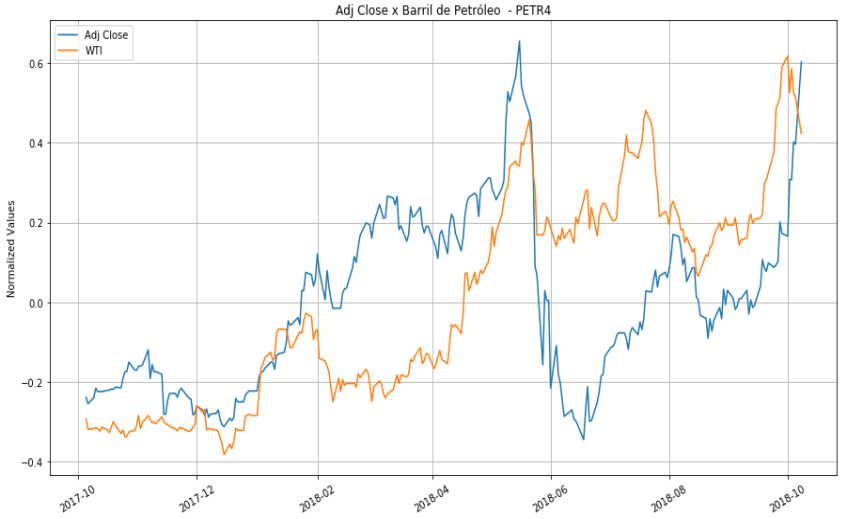


Figura 21 - Adj Close x WTI

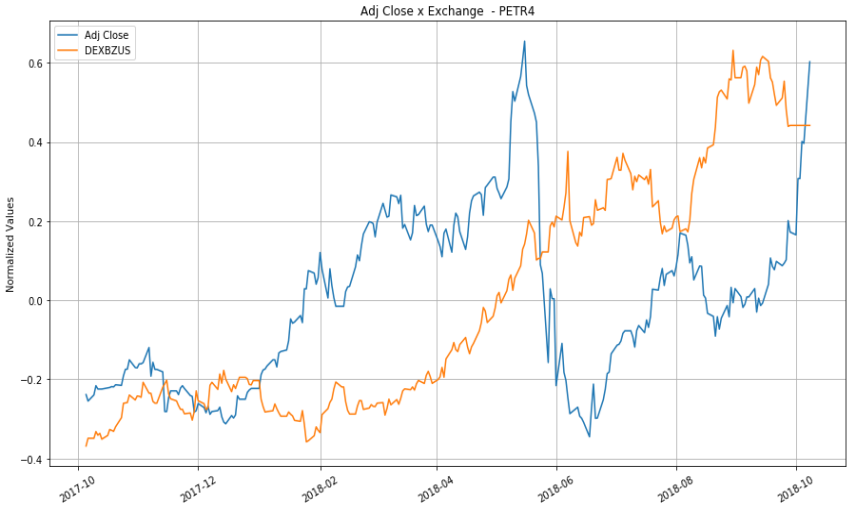


Figura 22 - Adj Close x Exchange

As figuras 20, 21 e 22 ilustram o que foi citado acima para a PETROBRÁS, onde a variação cambial e o preço do Barril de Petróleo não apresentam qualquer influência no preço da ação com a alta brusca da volatilidade (risco). Essa visualização não é apresentada explicitamente dentro sistema, mas foi verificada no projeto e comissionamento do mesmo

Melhorias

Algumas melhorias podem ser implementadas no sistema em conjunto com algumas técnicas de *deep learning* (redes neurais) para a partir dos resultados preditos o sistema sugerir boas ações para compra e venda para *traders* de curto, médio e longo prazo.

Partindo desse ponto, o sistema poderia através de um algoritimo de *Long & Short* por exemplo, manter um portfólio de ações para realizar operações de compra e venda casada afim de obter lucro.

Outra melhoria poderia ser a inserção de análise para derivativos, como opções, futuros e etc. Estes derivativos são intimanente ligados à *Volatilidade* , preços diretamente proporcionais à variação da volatilidade. O sistema poderia gerar recomendações de operações estruturadas com derivativos de acordo com o resultado da comparação da sua *volatilidade Implícita* e *a Volatilidade Gardch(1,1)* do ativo associado. Esta implementação requer um estudo mais profundo, principalmente no modelo matemático de precificação destes derivativos e da *Volatilidade Implícta*.

Referências

[1] *https://en.wikipedia.org/wiki/AdaBoost*

[2] *https://towardsdatascience.com/gradient-descent-in-a-nutshell-eaf8c18212f0*