Machine Learning Engineer Nanodegree

Projeto final

Marcos Wilson Sarinho Brito
22 de junho de 2019

I. Definição

Visão geral do projeto

Dentre todas as opções de investimentos, com certeza o investimento em ações é um dos que oferecem os melhores retorno e, ao mesmo tempo, um dos mais complexos. Historicamente os investidores utilizam as mais variadas fontes de informações, ferramentas e gráficos complexos buscando antever o movimento das ações.

Conforme as técnicas de IA se tornaram mais acessíveis, elas começaram a ser utilizadas também no mundo dos investimentos, buscando auxiliar a investidores.

Descrição do problema

Há vários estudos acadêmicos sobre a utilização de AI para gerenciamento de investimentos, cada um com um foco diferente, podemos citar estudos em gestão de carteiras[1], backtesting[2] e claro, previsão de preços, utilizando as mais diferentes técnicas[3].

O proposito deste projeto é desenvolver uma PoC (Proof of Concept) de uma ferramenta baseada em AI que possa antever as movimentações de ações (alta ou baixa) de empresas brasileiras, utilizando apenas dados básicos de negociações (valores de abertura e fechamento, mínimas e máximas, etc.).

O projeto será composto pelas seguintes tarefas:

- Obtenção de dados: download dos dados necessários para o projeto;
- Exploração de dados: verificação da integridade dos dados, formato e distribuição;

[1] http://discovery.ucl.ac.uk/1474136/1/PhDThesis_ToyinAwoye.pdf
[2] https://faculty.fuqua.duke.edu/~charvey/Research/Published_Papers/SSRN-id3275654.pdf
[3] http://cs229.stanford.edu/proj2017/final-reports/5212256.pdf

- Pré-processamento: preparação dos dados, incluindo, excluindo, normalizando e transformando os dados de modo a maximizar a leitura pelo algoritmo;
- Implementação do modelo: utilizar os dados para implementar diferentes modelos e selecionar o de melhor desempenho;
- Tuning do modelo: após selecionar o melhor modelo, fazer o refinamento do hiperparametros, encontrando valores que melhorem o desempenho do modelo

Depois de realizadas estas tarefas teremos um modelo pronto para fazer predições, é esperado que este modelo seja significativamente mais preciso que o benchmark.

Métricas

A métrica utilizada para este projeto será a precisão com que o algoritmo prevê as altas, seguindo a formula abaixo:

$$\text{PRECISÃO} = \frac{\text{verdadeiro positivo}}{\text{verdadeiro positivo + falso positivo}}$$

Esta foi considerada a métrica mais adequada para medir a eficiência do modelo porque consideramos que modelo causará um prejuízo financeiro maior ao falhar na indicação de uma alta do que se errar ao indicar uma queda. Em outras palavras: é melhor vender uma ação e ela e o valor continuar subindo do que continuar com ela e o valor cair.

II. Análise

Exploração dos dados

Para este projeto foram utilizados dados de negociações de ações da Cemig(CMIG4), Itaú(ITUB4), Magazine Luiza(MGLU3) e Petrobras(PETR4), empresas escolhidas aleatoriamente, na bolsa de valores de São Paulo obtidos no site Yahoo Finance[4].

Inicialmente pretendia-se utilizar os dados referentes aos últimos 8 anos, porém após uma primeira verificação, notou-se que os datasets estavam incompletos, por exemplo:

[4] http://finance.yahoo.com

'PETR4'

'Numero de registros: 1986'

	Open	High	Low	Close	Adj Close	Volume
count	1957.000000	1957.000000	1957.000000	1957.000000	1957.000000	1.957000e+03
mean	16.966725	17.228723	16.685948	16.942938	15.754821	4.491398e+07
std	5.397946	5.440303	5.351579	5.396971	5.030811	3.177863e+07
min	4.200000	4.270000	4.120000	4.200000	4.008060	0.000000e+00
25%	13.300000	13.590000	13.070000	13.290000	12.565537	2.540160e+07
50%	17.350000	17.660000	17.049999	17.299999	15.894867	3.870030e+07
75%	20.670000	21.080000	20.400000	20.670000	18.848883	5.649520e+07
max	29.549999	29.600000	28.980000	29.250000	29.119148	6.989506e+08

	Date	Open	High	Low	Close	Adj Close	Volume
41	2011-07-29	NaN	NaN	NaN	NaN	NaN	NaN
42	2011-08-01	NaN	NaN	NaN	NaN	NaN	NaN
43	2011-08-02	NaN	NaN	NaN	NaN	NaN	NaN
44	2011-08-03	NaN	NaN	NaN	NaN	NaN	NaN
45	2011-08-04	NaN	NaN	NaN	NaN	NaN	NaN
46	2011-08-05	NaN	NaN	NaN	NaN	NaN	NaN
47	2011-08-08	NaN	NaN	NaN	NaN	NaN	NaN
48	2011-08-09	NaN	NaN	NaN	NaN	NaN	NaN

Devidos aos papeis selecionados serem de grande liquidez, causou estranheza esta falta de informações, então verifiquei em outras fontes (sites da B3[5] e Advf[6]) que as informações de fato existiam, só não estavam no dataset do Yahoo Finance.

Foi avaliada a possibilidade de trocar o fornecedor dos dados, entretanto uma das fontes não fornece o layout do arquivo(B3) e a outra fonte não tem uma opção gratuita de exportação de arquivos.

Por fim, optou-se por diminuir o alcance do dataset, uma vez que os registros com problemas se concentram nos anos de 2011-2013. Portanto foram utilizados os dados no intervalo de 01/01/2014 a 01/06/2019.

Segue descrição do dataset:

Date (data base),

Open (valor de abertura na data base),

High (valor máximo na data base na data base),

Low (valor mínimo na data base),

Close (valor de fechamento na data base),

[5] http://www.bmfbovespa.com.br/en_us/services/market-data/historical-data/equities/historical-data/
[6] https://br.advfn.com/bolsa-de-valores/bovespa/petrobras-PETR4/historico/mais-dados-historicos

Adj Close (percentual de ajuste na data base),

Volume (volume de operações na data base),

Predict (indica alta ou baixa) *

*A feature “Adj Close” foi utilizada para gerar a coluna “Predict” (feature target), onde o valor 1 indica uma alta em relação ao dia anterior e o valor 0 indica uma baixa em relação ao dia anterior.

Além dos dados de negociação, também foram selecionados alguns índices de mercado para serem incluídos no modelo:

- Selic[7]
- IPCA[8]
- IPCA15[9]
- IBC-BR[10]
- PIB[11]
- PTAX[12]
- IBovespa[13]

Embora todos estes índices sejam relevante e possam aprimorar a precisão do modelo, apenas os índices Selic, Dólar PTAX e Ibovespa foram utilizados. Os outros foram descartados por não informar a data de divulgação, os datasets tinham apenas a data de referência. Partindo do princípio que a divulgação destes índices gera reflexos nas ações de modo imediato, utiliza-los apenas com a data de referência iria inserir uma informação de covariância incorreta no modelo. Por exemplo: se o índice IPCA (índice de inflação) impacta o valor das ações da empresa X, o impacto será sentido no dia da divulgação do índice e não no início ou final do mês.

O dataset com os índices tem as seguintes informações:

Date (data),

IBOV (índice Ibovespa),

Dolar_PTAX (cotação do dólar PTAX),

---

[7] https://www.bcb.gov.br/controleinflacao/historicotaxasjuros
[8] https://www.ibge.gov.br/estatisticas/economicas/precos-e-custos/9260-indice-nacional-de-precos-ao-consumidor-amplo-15.html?=&t=downloads
[9] https://www.ibge.gov.br/estatisticas/economicas/precos-e-custos/9256-indice-nacional-de-precos-ao-consumidor-amplo.html?=&t=series-historicas
[10]https://www3.bcb.gov.br/sgspub/consultarvalores/consultarValoresSeries.do?method=consultarGraficoPorId&hdOidSeriesSelecionadas=24363
[11]https://www.bcb.gov.br/estatisticas/indicadoresconsolidados
[12]https://www3.bcb.gov.br/sgspub/consultarvalores/consultarValoresSeries.do?method=consultarGraficoPorId&hdOidSeriesSelecionadas=1
[13] http://finance.yahoo.com

Selic (taxa Selic divulgada pelo Copom) **

*Os índices são fornecidos por diferentes fontes e em diferentes formatos, contudo, eles foram agrupados num único dataset.

**A taxa Selic é divulgada com periodicidade aproximada de um mês

Visualização exploratória

Segue abaixo algumas estatísticas dos dados de negociação do MGLU3

```
'MGLU3'
'Numero de registros: 1346'
```

	Open	High	Low	Close	Adj Close	Volume	Predict
count	1346.000000	1346.000000	1346.000000	1346.000000	1346.000000	1.346000e+03	0.0
mean	46.972052	47.863819	45.999954	46.961683	46.458773	1.264631e+06	NaN
std	58.586739	59.580154	57.492897	58.557629	58.507392	1.076972e+06	NaN
min	1.006250	1.016250	0.972500	0.978750	0.929990	0.000000e+00	NaN
25%	5.432500	5.530000	5.331563	5.457813	5.112449	6.323250e+05	NaN
50%	9.347500	9.510000	9.190625	9.396250	8.657102	1.002350e+06	NaN
75%	80.190000	81.327501	78.220000	79.600003	78.843594	1.588800e+06	NaN
max	201.500000	202.309998	194.029999	200.210007	200.210007	1.345200e+07	NaN

Note que, exceto a coluna “Predict”, todas as outras colunas tem o mesmo número de registros (count) que o dataset, ou seja, todas as linhas tem informações em todas as colunas.

Abaixo podemos ver uma demonstração dos dados:

	Date	Open	High	Low	Close	Adj Close	Volume	Predict
0	2014-01-02	7.61	7.82	6.95	7.23	6.493703	1278000	NaN
1	2014-01-03	7.30	7.49	7.07	7.47	6.709262	802700	NaN
2	2014-01-06	7.50	7.90	7.49	7.86	7.059545	599700	NaN
3	2014-01-07	7.90	8.27	7.90	8.06	7.239178	1253800	NaN
4	2014-01-08	8.04	8.24	8.01	8.22	7.382883	791000	NaN
5	2014-01-09	8.18	8.47	8.03	8.18	7.346957	1361800	NaN
6	2014-01-10	8.26	8.37	8.08	8.15	7.320010	1043200	NaN
7	2014-01-13	8.21	8.26	8.03	8.20	7.364920	648700	NaN
8	2014-01-14	8.16	8.33	8.07	8.25	7.409827	541800	NaN
9	2014-01-15	8.25	8.40	8.15	8.28	7.436772	502900	NaN

Como trata-se de dados financeiros, todos os valores são numéricos, sem textos.

Abaixo temos uma demonstração dos dados de índices:

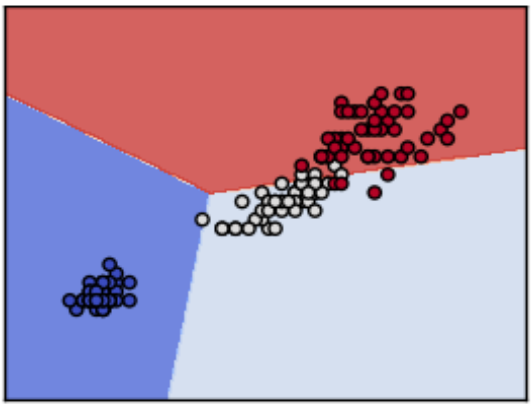
	Date	IBOV	Dolar_PTAX	Selic
0	2014-01-02	50341.0	2.3975	10.0
1	2014-01-03	50981.0	2.3741	10.0
2	2014-01-06	50974.0	2.3789	10.0
3	2014-01-07	50430.0	2.3634	10.0
4	2014-01-08	50577.0	2.3779	10.0
5	2014-01-09	49322.0	2.3960	10.0
6	2014-01-10	49696.0	2.3819	10.0
7	2014-01-13	49427.0	2.3491	10.0
8	2014-01-14	49703.0	2.3617	10.0
9	2014-01-15	50105.0	2.3470	10.5

Algoritmos e técnicas

Foram testados os algoritmos LinearSVC, KNN e AdaBoost, pois cada um utiliza uma lógica de funcionamento diferente, assim pudemos garantir que encontramos a melhor solução para o problema, segue abaixo uma descrição resumida do funcionamento deles:

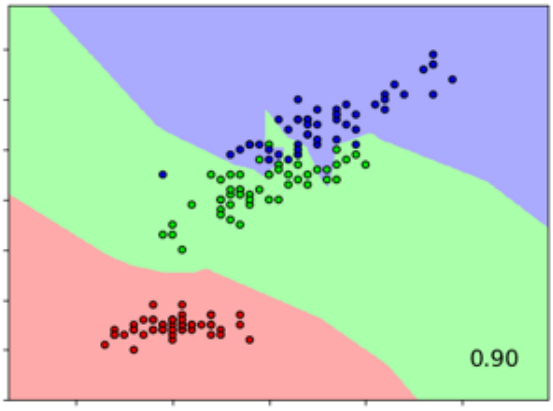
Linear SVC

O LinearSVC funciona tentando desenhar uma linha entre instâncias e maximizando a largura das linhas para que possamos identificar facilmente se uma instância pertence à classe A ou B. Ele funciona surpreendentemente bem se seus dados são linearmente separáveis, mas a maioria dos dados infelizmente não é.



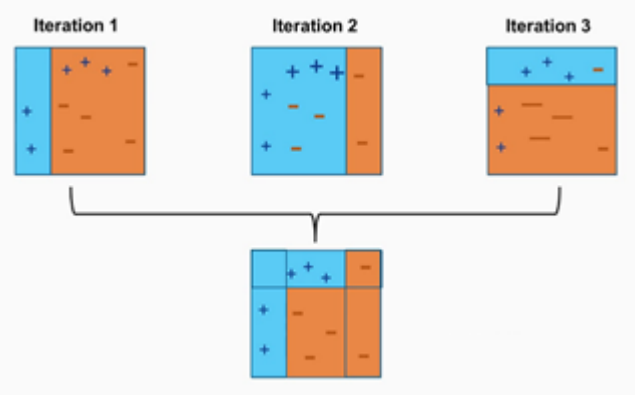
KNN

O algoritmo KNN assume que coisas semelhantes existem em estreita proximidade, em outras palavras, coisas semelhantes estão próximas umas das outras. Partindo desta logica, ele calcula a distância dos K vizinhos próximos para fazer a previsão. O KNN é automaticamente não-linear, ele pode detectar dados distribuídos lineares ou não-lineares, ele tende a ter um desempenho muito bom com muitos pontos de dados.



AdaBoost

AdaBoost é um método de conjunto geral que cria um classificador forte a partir de vários classificadores fracos. Isso é feito construindo um modelo a partir dos dados de treinamento e criando um segundo modelo que tenta corrigir os erros do primeiro modelo. Os modelos são adicionados até que o conjunto de treinamento seja previsto perfeitamente ou um número máximo de modelos seja adicionado.



Conforme esperado, o KNN apresentou o melhor desempenho, uma vez que dados de ações não tem distribuição linear.

Benchmark

Para esta PoC usaremos um modelo simples de benchmark, visando apenas sinalizar altas e baixas, seguindo a lógica do benchmark utilizado no paper Stock Market Forecasting Using Machine Learning Algorithms[14].

O modelo irá assumir que o ajuste no fechamento de amanhã é maior que hoje caso o de hoje tiver sido maior que o de ontem. Este modelo terá uma boa performance enquanto o mercado seguir as tendências do dia anterior e terá baixo desempenho quando tivermos inversões no mercado.

III. Metodologia

Pré-processamento de dados

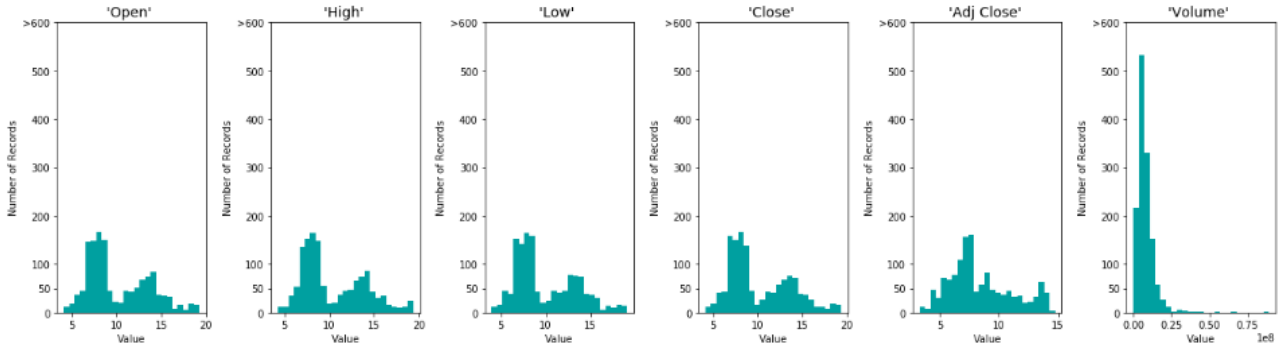
Conforme documentado na seção Exploração dos Dados, antes mesmo de fazer o pré-processamento, foi necessário alterar o intervalo de tempo da captura das informações. Embora fosse possível utilizar uma metodologia especifica para completar os dados de negociação, optei por não utilizar por considerar que estaria inserindo dados irreais no dataset.

[14]https://pdfs.semanticscholar.org/b68e/8d2f4d2c709bb5919b82effcb6a7bbd3db37.pdf

O pré-processamento foi teve as seguintes etapas:

- Geração coluna target: nesta etapa são gerados os dados para a coluna “Predict”. A função simplesmente calcula o valor se o valor “Adj Close” de D+0 é maior ou menor que D+X, onde X é o número de dias da previsão. Caso D+0 for maior que D+X, isso indica uma queda e o de “Predict” é preenchido com o valor 0, caso contrário, indica uma alta e o campo “Predict” é preenchido com o valor 1.

- Transformação logarítmica: esta etapa é importante quando os valores de uma feature tem uma variação muito grande, por exemplo: volume de vendas num dia normal x vendas no natal, pois estes picos de distribuição podem influenciar alguns tipos de algoritmos. Nesta etapa primeiro apresentamos gráficos com as distribuições das features com a finalidade de identificar problemas de distribuição. Como apenas os dados de Volume apresentaram grande pico de distribuição, a transformação logarítmica foi aplicada apenas a esta feature.



- Normalização: esta técnica é aplicada nos dados numéricos para transforma-los em uma escala pré-definida, evitando que o algoritmo aplique pesos diferente apenas pelo fato das diferentes colunas trabalharem com escalas diferente (ex.: enquanto os campos de valores de ações costumam ficar na casa das dezenas, o campo de volume frequentemente fica na casa dos milhares).

- Divisão de dados de treinamento e teste: aqui é definida a coluna “Predict” como o alvo e os dados de são divididos em dois grupos, treino e teste. Esta divisão é feita para que possamos testar o algoritmo com uma massa de dados que ele não interagiu anteriormente, o que facilita validar a eficácia e a existência de overfitting.

Implementação

O primeiro passo foi o download dos dados de negociações (em formato csv) e a leitura dos arquivos, de modo a permitir a exploração dos dados. Devido ao problema explicado anteriormente na seção Exploração de Dados, foi necessário fazer novos downloads e leitura de arquivos, agora utilizando um intervalo menor de tempo.

O próximo passo, Pré-processamento, consistiu em escrever as funções necessárias para cada etapa descrita na seção Pré-processamento de dados (funções pre_processing_target_column, pre_processing_verify_log_transformed, pre_processing_log_transformed, pre_processing_minmax_transform, pre_processing_sort_training_test).

Após aplicar estas funções no dataset da empresa selecionada, tínhamos como saída os conjuntos de dados de treinamento e teste prontos para serem utilizados nos algoritmos de inteligência artificial selecionados.

Em seguida foram escritas as funções para treinar e comparar os modelos, que logo depois foram aplicadas aos dados já tratados e divididos, obtendo como saída os dados de desempenho dos modelos testados, permitindo que façamos a escolha do melhor.

Para finalizar, foram aplicadas técnicas de tuning no modelo de melhor desempenho, visando verificar a possibilidade de aumentar o desempenho deste.

Neste ponto, com o resultado do desempenho do modelo em mãos, decidiu-se incluir também no dataset os dados de alguns índices de mercado, visando a melhoria do desempenho do modelo. Para isso foram selecionados alguns índices, baixados os arquivos e feita a exploração dos dados. Algumas dificuldades foram encontradas, conforme explicado anteriormente na seção Exploração de Dados.

Por fim, os dados dos índices passiveis de serem utilizados foram inseridos no dataset de negociações, passando por todas as etapas de pré-processamento e novamente foram aplicados os modelos. Ao fim deste processo, deu-se por concluída a implementação

Refinamento

O tuning do modelo foi realizado com o apoio do GridSearchCV, esta função é muito útil quando precisamos testar vários valores diferentes para os hiperparâmetros do modelo em busca de uma melhor performance.

Após identificarmos que o modelo KNN apresentava a melhor performance, o GridSearchCV foi utilizado para identificar quais seriam os melhores valores para os hiperparâmetros abaixo descritos:

- leaf_size: Tamanho da folha passado para o BallTree ou KDTree. Isso pode afetar a velocidade da construção e consulta.
- n_neighbors: Número de vizinhos a usados por padrão para consultas do KNN.
- p: Parâmetro de potência para a métrica Minkowski. Quando p = 1, isso equivale a usar manhattan_distance (l1) e p=2 para euclidean_distance (l2).

Segue tabela os valores destes hiperparametros:

Parâmetro	Default	Otimizado
leaf_size	30	1
n_neighbors	5	3
P	2	1

IV. Resultados

Modelo de avaliação e validação

O algoritmo escolhido foi o KNN, pois foi o que apresentou a melhor precisão nos testes realizados, tanto com o dataset com dados de negociação quanto com o dataset com dados de negociação mais índices de mercado.

Foram realizados testes com quatro empresas, de diferentes setores (energia, financeiro, varejo e petróleo), de modo a garantir que o modelo está generalizando para dados totalmente diferentes sem grandes variações na precisão.

Após a otimização, o modelo chega a um nível de precisão entre 73 e 76, variando conforme os dados. Considero que esta precisão é boa, uma vez que é bem superior ao benchmark, porém o nível de confiabilidade ainda está baixo para o tipo de atividade que se propõe. Embora possa ser uma ferramenta interessante para investidores com pouco conhecimento, para profissionais seria de pouca utilidade.

Entretanto, devemos observar atentamente o ganho de performance obtido coma inclusão de índices de mercado no dataset de negociações, me parece crível que o modelo pode melhorar ainda mais caso novos índices sejam incluídos.

Justificativa

A princípio, os resultados obtidos apenas utilizando os dados de negociação não foram muito animadores, embora melhor que os valores de referência, ainda tinham uma precisam baixa. Porém com a inclusão dos índices no dataset, houve um ganho significativo, ficando próximo de uma precisão mínima aceitável.

Segue abaixo tabela com o índice de precisão do modelo:

	CMIG4	ITUB4	MGLU3	PETR4
Benchmark	0.4813	0.494	0.5172	0.5149
KNN – Dados de negociação	0.6198	0.6333	0.6108	0.585
KNN – Dados de negociação + índices	0.7425	0.7528	0.6894	0.715

V. Conclusão

Forma livre de visualização

Abaixo podemos ver alguns gráficos preparados durante o desenvolvimento do projeto. Na primeira sequência de gráficos podemos ver o desempenho dos diferentes algoritmos antes e depois da inclusão dos índices.

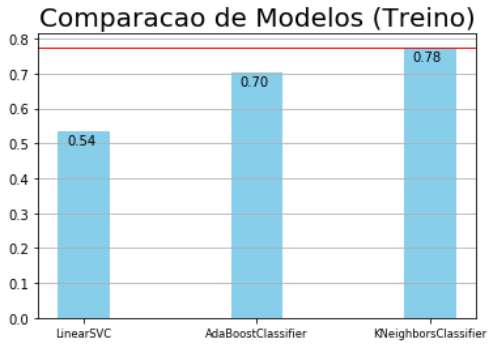
Alguns pontos interessantes que podemos notar:

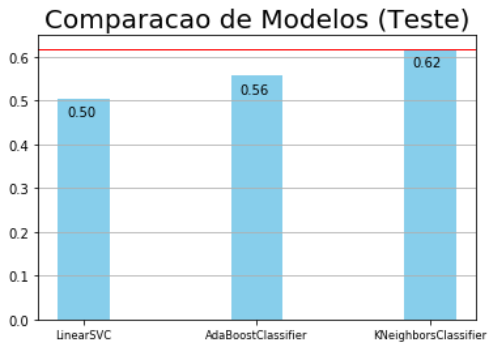
1. O algoritmo KNeighborsClassifier só foi superado por outro algoritmo no gráfico da empresa MGLU3, com dados de teste e antes de incluir os novos índices.

2. O algoritmo LinearSVC apresentou a pior precisão em todos os gráficos, definitivamente não deve ser utilizado para este tipo de problema.

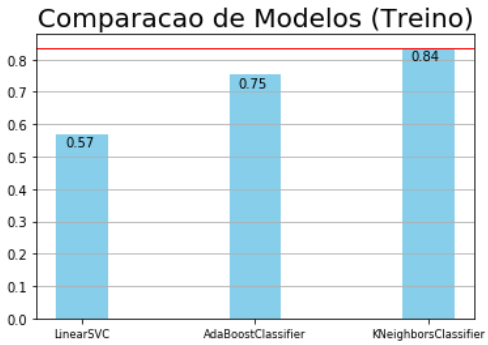
3. As previsões de todas as ações se beneficiaram da inclusão dos índices de mercado, sendo que a maioria apresentou melhora superior a 20%.

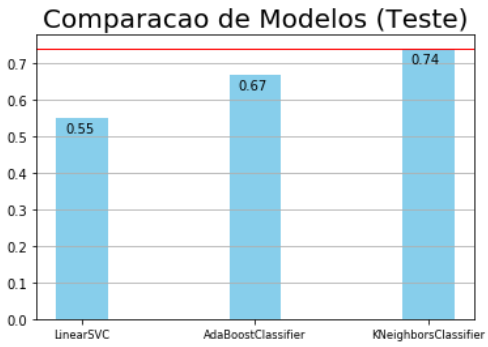
CMIG4



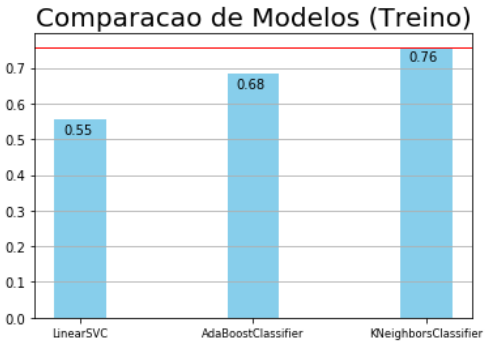


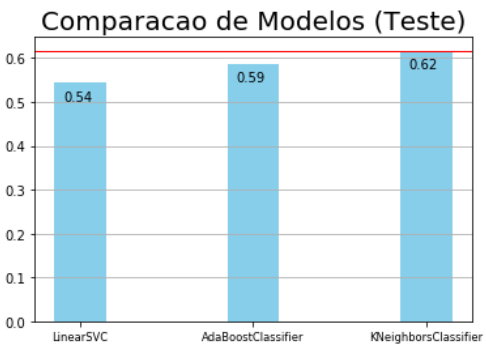
Inserindo Índices de mercado



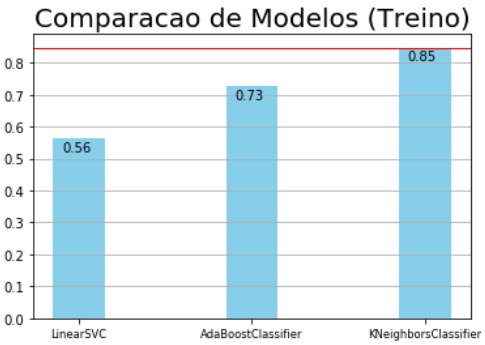


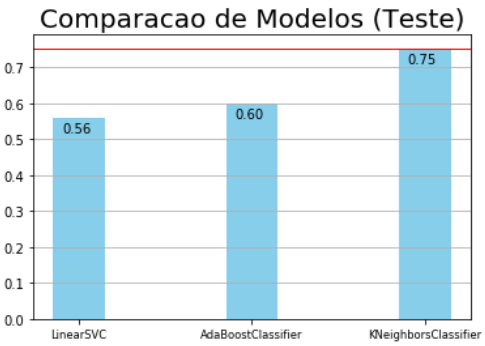
ITUB4



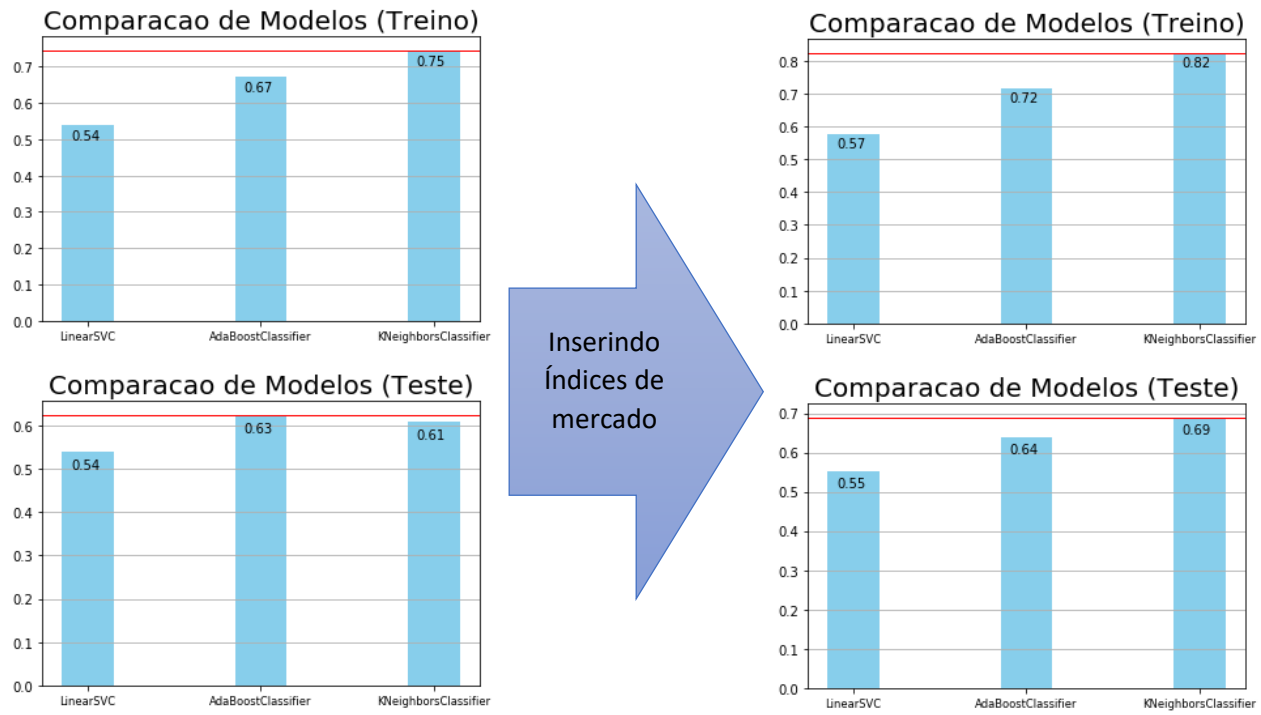


Inserindo Índices de mercado

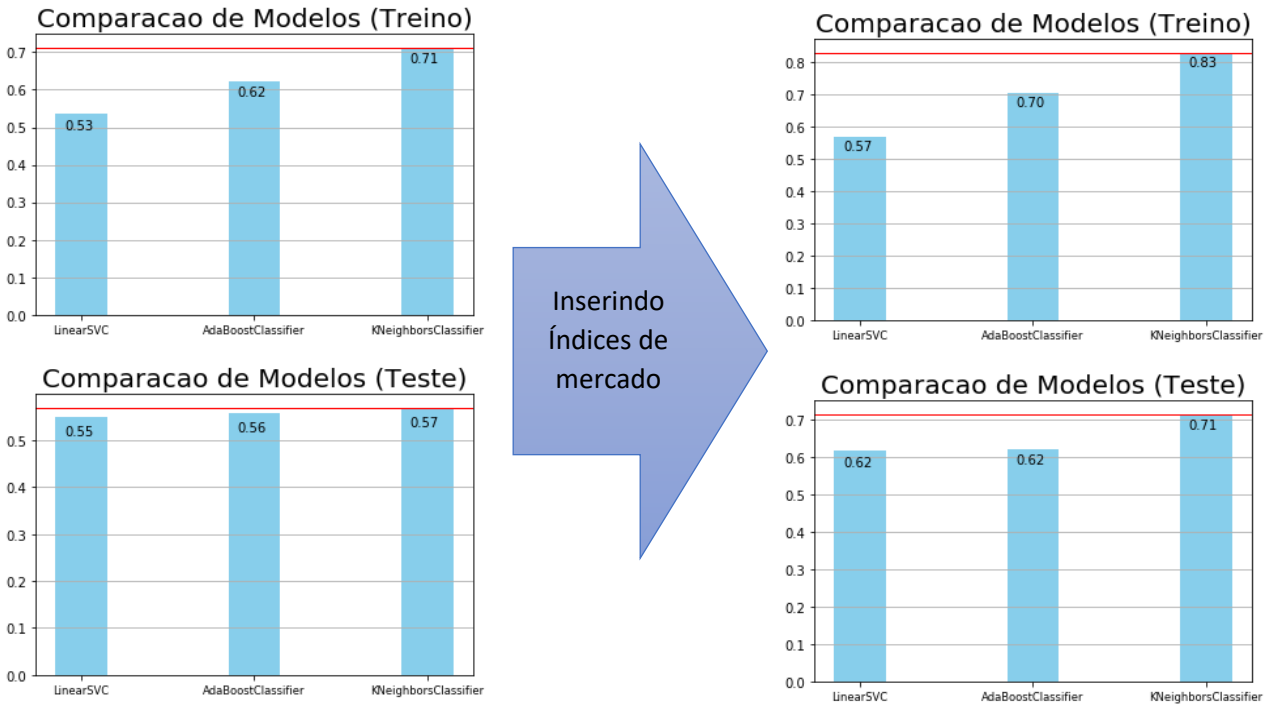




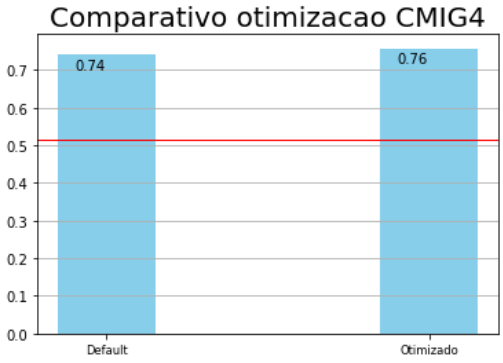
MGLU3

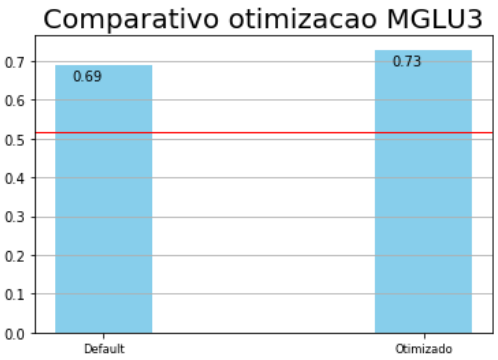


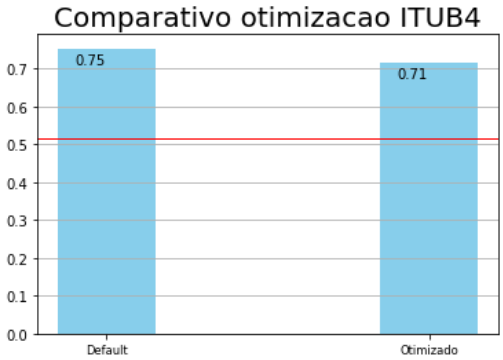
PETR4

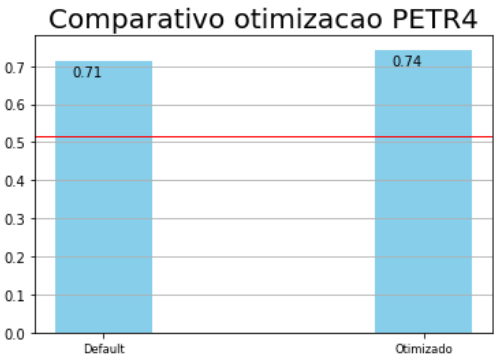


E por ultimo fechamos o projeto com a otimização do modelo, conforme gráficos abaixo, nesta etapa conseguimos melhorar um pouco mais o desempenho:









*inesperadamente a precisão para a empresa ITUB4 são apresentadas como piores após a otimização. Analisando os hiperparametros utilizados para o ITUB4, notou-se que foi o único caso em que foi configurado `n_neighbors=1`. Se faz necessária uma investigação mais profunda para entender porque a biblioteca causou este, de qualquer forma este problema não causa preocupação, uma vez que basta utilizar o modelo mais eficiente e testar outras opções manualmente.

Reflexão

O primeiro passo do projeto, a obtenção dos dados de negociação, acabou por ser mais desafiador do que o esperado. Devido a seleção de empresas de grande porte, a expectativa era que os dados de negociação estivessem íntegros, sendo a exploração deles apenas uma formalidade para confirmar status, porém após a primeira analise notou-se problemas de falta de dados, o que demandou uma segunda verificação na interface web e depois a validação junto a outras fontes de dados. Confirmada a ausência de alguns registros, tomou-se a decisão de alterar o intervalo de tempo dos dados, afim de evitar a troca de fornecedor de dados.

Superado o obstáculo inicial da obtenção dos dados e exploração, o projeto fluiu mais tranquilamente, sendo as etapas de pré-processamento, implementação do modelo, validação e tuning relativamente fáceis de implementar com a utilização do sklearn, que fornece muitas funcionalidades que realmente agilizam o processo. Destaco, porém, a importância de todas as etapas, durante o processo eu esqueci de aplicar a transformação logarítmica em algumas features, isto causou uma grande distorção, fazendo a precisão com o KNN cair muito, a ponto do AdaBoost ter melhor desempenho.

A fase seguinte foi implementar uma função de previsão para ser utilizada para benchmark, encontrar um paper indicando um exemplo de benchmark foi mais demora que a implementação em si.

Após fazer as implementações, chegou a hora de ver os resultados e eles não foram nada bons: embora o algoritmo escolhido tivesse uma precisão superior ao benchmark, ainda era muito pouco. De certa forma isso respondia a proposta da PoC: um modelo baseado apenas em dados básicos de negociação não tem como fazer previsões muito acuradas.

Com este mal resultado em mãos, partimos para a melhoria mais promissora: inserir índices de mercado no modelo. Por serem informações públicas, considero que continuamos seguindo a proposta do projeto.

Todavia, inserir mais informações no dataset nos levou de novo ao primeiro passo do projeto: download de arquivos e exploração de dados. Embora todos os índices sejam informações públicas, amplamente divulgados, quando precisamos fazer download deles nos deparamos com a realidade: diferentes índices de diferentes institutos, em sites diferente e layout de arquivo diferentes. Contudo, isto não foi o maior problema, o que realmente causou dor de cabeça foram alguns índices importantes não terem a data de divulgação, o que impossibilitou a utilização deles.

De qualquer forma, seguimos conforme o planejado, utilizando os índices que eram possíveis, fazendo um pré-processamento pontual e unindo as novas informações aos dados de negociação.

Agora com um novo dataset em mãos, bastou aplicas novamente todas as etapas que haviam sido feitas anteriormente (pré-processamento, implementação do modelo, validação e tuning). A inclusão de novos dados apresentou o resultado esperado, uma melhora na precisão do modelo.

Acredito que a implementação do projeto exemplificou o que temos de problemas em projetos reais: a dificuldade de se obter grandes massas de dados confiáveis por vezes pode ser maior que a implementação do resto do projeto, onde podemos contar com boas ferramentas, como o sklearn.

Melhorias

Há uma série de melhorias que podem ser implementadas e certamente irão melhorar o desempenho e a confiabilidade dos resultados. As melhorias foram divididas em dois tipos: dados e técnicas.

Melhorias de dados:

1. Mais volume de dados: seria interessante viabilizar a utilização de dados de um intervalo de tempo maior.

2. Inclusão de mais índices: seria importante inserir os índices econômicos básicos, como IPCA e IBC-BR. Também seria muito interessante verificar a covariância junto com dados do relatório Focus, embora seja difícil encontrar uma fonte que forneça estes dados tabulados e formatados.

3. Indicadores de análise gráfica: aqui já estamos saindo do proposito inicial de basear-se apenas nos dados brutos, mas caso haja a vontade de elevar o desempenho a outro nível, a inclusão de indicadores técnicos (ex.: bandas de Bollinger) certamente precisam estar presentes.

Melhorias técnicas:

1. Utilizar KFold para validação[15]: uma melhoria importante é começar a utilizar um método de cross validation no lugar do train/test split. Utilizando KFold, teremos menos problemas causados por viés.

2. Tuning com RandomizedSearchCV[16]: uma vez que aumentarmos o volume de dados, fazer o tuning do modelo com RandomizedSearchCV ajudará a ganhar tempo, sem perda de funcionalidade em relação ao GridSearchCV.

3. Modelos mais complexos: foram utilizados modelos clássicos, largamente utilizados e conhecidos há um bom tempo, porém temos modelos mais recentes, como LGBM[17] e XGBOOST[18], que tem apresentado ótimos resultados e merecem ser avaliados.

4. Usar ferramentas para explicar a predição: conforme inserirmos mais e mais features ao nosso conjunto de dados, torna-se necessário utilizar bibliotecas como o LIME[19] e Shap[20] nos ajudar a explicar quais features de fato estão influenciando o nosso modelo. Explicar a predição pode ser um grande diferencial par a ferramenta.

5. Ferramentas visuais: o projeto faz pouco uso de recursos visuais, seria interessante dedicar um tempo para implementar apresentações utilizando Seaborn [21]ou Plotly[22]. Caso surja a necessidade de apresentar as cotações em tempo real, vale a pena dedicar um tempo a biblioteca Bokeh[23], que trabalha com streaming.

[15] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html
[16] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html
[17] https://lightgbm.readthedocs.io/en/latest/_modules/lightgbm/sklearn.html
[18] https://machinelearningmastery.com/develop-first-xgboost-model-python-scikit-learn/
[19] https://www.analyticsvidhya.com/blog/2017/06/building-trust-in-machine-learning-models/
[20] https://shap.readthedocs.io/en/latest/
[21] https://seaborn.pydata.org/
[22] https://plot.ly/python/
[23] https://zduey.github.io/snippets/streaming-stock-data-with-bokeh/