Usando notícias para prever o movimento de ações

Felipe Lana Machado[1]

[1]Engenharia de Software – Universidade do Estado de Santa Catarina (UDESC)
R. Dr. Getúlio Vargas, 2822 - Bela Vista, Ibirama - SC, 89140-000

Abstract. *This article introduces Udacity's Machine Learning Engineer course completion project. The first is a reference of the subject, followed by an initial analysis of the data and the detailed description of the problem. Then a scan is performed on the data, performing all necessary adjustments. Finally, the result is discussed and analyzed, as well as compared to the reference model used.*

Resumo. *Este artigo apresenta o projeto de conclusão de curso de Engenheiro de Machine Learning da Udacity. Sendo que primeiro é feita uma referência do assunto, seguido por uma análise inicial dos dados e a descrição detalhada do problema. Em seguida é realizada uma exploração nos dados, realizando todas as adequações necessárias. Por fim, é discutido e analisado o resultado, bem como comparado com o modelo de referência utilizado.*

1. Histórico do Assunto

O mercado financeiro sempre foi alvo de muitos estudos, sempre houveram diversas formas de se analisar o mercado afim de tomar a melhor decisão para obter o melhor rendimento. Muitos acreditam que o mercado é influenciado por noticias e acontecimentos que ocorrem diariamente, essas noticias podem ser tanto positivas ou negativas quanto neutras para um determinado ativo na bolsa de valores.

Com o aumento da performance dos hardwares, conseguindo processamentos cada vez mais velozes por um menor custo, aliado a um volume de dados crescente e cada vez de melhor qualidade, a área de pesquisa do aprendizado de máquina tem cada vez mais se tornado central na tentativa de melhorar o desempenho em praticamente todas as atividades lucrativas.

Desta forma, aliando o crescente volume de noticias relevantes que possam impactar o mercado financeiro com o aumento da capacidade de processamento dos computadores, tem-se como objetivo encontrar um modelo que permita verificar quais noticias são relevantes para prever o movimento das ações. E com isso conseguir prever o movimento do mercado financeiro utilizando noticias.

2. Descrição do problema

Através da competição do Kaggle [Kaggle 2018] que define o objetivo predizer como as ações irão mudar baseados no estado do mercado e novas notícias. Para isso, é necessário respeitar algumas regras como não utilizar dados futuros para realizar predições do dia atual de treinamento. Ou seja, não se pode utilizar dados futuros no treinamento para predizer algo no ”passado”.

Outras regras também foram impostas a competição a fim de minimizar as discrepâncias dos resultados, assim como tentar excluir resultados obtidos de forma ilegal

para o projeto. Assim, para o desenvolvimento do projeto tem como restrições o tempo de execução do algoritmo que não pode ser superior a 6 horas, além disso não se pode utilizar a GPU para o processamento e nem possuir conexão com a internet, para que não sejam utilizados dados fora os disponibilizados. Também, não poderá passar de 16Gb consumidos durante a execução para que seja padronizado os recursos disponíveis entre os competidores.

3. Conjuntos de dados e entradas

Existem duas fontes de dados neste problema, a primeira são os dados do mercado financeiro (a partir de 2007) que é provida pelo Intrinio [Intrinio: Data Intelligence, on demand 2018]. Estes dados contém informações financeiras como preço de abertura, preço de fechamento, volume negociado, retorno calculado, etc. Enquanto a outra fonte de dados se refere as notícias (a partir de 2007), contém informações como artigos e alertas publicados sobre os ativos, detalhes do artigo, sentimento e outros comentários. Estes dados são fornecidos pela Thomson Reuters [Thomson Reuters 2017].

Os dados do mercado financeiro incluem um conjunto de ativos americanos, mostrando variações diárias. Por ser uma competição do Kaggle [Kaggle 2018], o autor informa que poderão haver valores ausentes nos dados e que apesar destes dados algumas vezes existirem, estes não poderão ser utilizados pois não estará de acordo com o critério de seleção dos dados escolhido.

- time(datetime64[ns, UTC]) - o horário corrente (nos dados do mercado,todas as linhas foram obtidas às 22:00 UTC)
- assetCode(object) - id único do ativo
- assetName(category) - o nome que representa um grupo de códigos de ativos. Pode ser desconhecido (”Unknown”) se o codigo do ativo correspondente não tiver em nenhuma linha nos dados de notícias.
- universe(float64) - um booleano indicando se o ativo, naquele dia, será incluído na pontuação ou não. O universo de ativos disponíveis em um determinado dia, são os ativos que estão disponíveis para transações. O universo de transações muda diariamente.
- volume(float64) - volume de transações em participações para o dia.
- close(float64) - o preço de fechamento para o dia (slipts e dividendos não estão ajustados)
- open(float64) - o preço de abertura no dia (slipts e dividendos não estão ajustados)
- returnsClosePrevRaw1(float64) - veja a explicação a baixo
- returnsOpenPrevRaw1(float64) - veja a explicação a baixo
- returnsClosePrevMktres1(float64) - veja a explicação a baixo
- returnsOpenPrevMktres1(float64) - veja a explicação a baixo
- returnsClosePrevRaw10(float64) - veja a explicação a baixo
- returnsOpenPrevRaw10(float64) - veja a explicação a baixo
- returnsClosePrevMktres10(float64) - veja a explicação a baixo
- returnsOpenPrevMktres10(float64) - veja a explicação a baixo

- `returnsOpenNextMktres10(float64)` - 10 dias, retorno-rezidualizado do mercado. Esta é a variável de alvo utilizada como pontuação na competição proposta. Os dados do mercado financeiro foram filtrados de forma com que returnsOpenNextMktres10 não seja nula jamais.

Quanto aos dados referentes as notícias, temos que os dados não estão normalizados propositalmente.

- `time(datetime64[ns, UTC])` - UTC horário que a notícia ficou disponível no feed (precisão de segundos)
- `sourceTimestamp(datetime64[ns, UTC])` - UTC horário em que foi criado o item de notícia
- `firstCreated(datetime64[ns, UTC])` - UTC horário da primeira versão do item
- `sourceId(object)` - um id para cada item de notícia
- `headline(object)` - o titulo do item
- `urgency(int8)` - tipo de história (1: alerta, 3: artigo)
- `takeSequence(int16)` - o número da sequencia para cada item de notícia, começando em 1. Para cada história, alertas e artigos tem sequencias separadas.
- `provider(category)` - identificação da organização que providenciou os itens de notícia (RTRS para Reuters News e BSW para Business Wire)
- `subjects(category)` - códigos de tópicos ou companhias que estão relacionadas ao item de notícia. Os tópicos em códigos descrevem o assunto do item de notícia, e podem cobrir classes, geografias, eventos, industrias/setores e outros tipos
- `audiences(category)` - identifica qual (is) produto (s) de notícias do desktop o item de notícias pertence. Eles são tipicamente adaptados para públicos específicos. (ex.”M”para Money International News Service e ”FB”para French General News Service)
- `bodySize(int32)` - o tamanho da atual história em caracteres
- `companyCount(int8)` - o número de companhias explicitamente listados nos itens de notícia nos campos de assunto
- `headlineTag(object)` - o Thomson Reuters título marcado para o item de notícia
- `marketCommentary(bool)` - booleano indicador que o item esta discutindo as condições gerais do mercado, como ”After the Bell”resumo.
- `sentenceCount(int16)` - o número total de sentenças no item de notícia. Pode ser utilizado em conjunto com firstMentionSentence para determinar a posição da primeira menção no item.
- `wordCount(int32)` - o número total de palavras e pontuações no item de notícia.
- `assetCodes(category)` - lista de ativos mencionados na notícia
- `assetName(category)` - nome dos ativos
- `firstMentionSentence(int16)` - a primeira frase, começando pelo título.
  - 1: titulo
  - 2: primeira frase no corpo
  - 3: segunda frase no corpo
  - 0: o ativo sendo avaliado não foi encontrado no titulo ou no corpo do texto. Assim, a notícia inteira será usada para avaliar o valor do sentimento.

- relevance(float32) - um valor decimal (0 a 1) indicando a relevância do item de notícia para o ativo financeiro. Se o ativo for mencionado no título, a relevância é 1. Quando for um alerta, a relevância deverá ser obtida através do firstMentionSentence.
- sentimentClass(int8) - indica a predominância do sentimento para esse item de notícia em relação ao ativo financeiro. A classe indicada é a que representa a maior probabilidade.
- sentimentNegative(float32) - probabilidade do sentimento no item de notícia ser negativo para o ativo.
- sentimentNeutral(float32) - probabilidade do sentimento no item de notícia ser neutro para o ativo.
- sentimentPositive(float32) - probabilidade do sentimento no item de notícia ser positivo para o ativo.
- sentimentWordCount(int32) - número de palavras ou pontuações que são relevantes para o ativo. Isso pode ser utilizado juntamento com o wordCount para verificar a proporção da notícia que está discutindo o ativo.
- noveltyCount12H(int16) - A novidade de 12 horas do conteúdo de um item de notícias em um recurso específico. É calculado comparando-o com o texto específico do ativo em um cache de itens de notícias anteriores que contêm o ativo.
- noveltyCount24H(int16) - o mesmo que acima, mas para 24 horas
- noveltyCount3D(int16) - o mesmo que acima, mas para 3 dias
- noveltyCount5D(int16) - o mesmo que acima, mas para 5 dias
- noveltyCount7D(int16) - o mesmo que acima, mas para 7 dias
- volumeCounts12H(int16) - o volume de notícias de 12 horas para cada ativo. Um cache de notícias anteriores é mantido e o número de itens de notícias que mencionam o ativo dentro de cada um dos cinco períodos históricos é calculado.
- volumeCounts24H(int16) - o mesmo que acima, mas para 24 horas
- volumeCounts3D(int16) - o mesmo que acima, mas para 3 dias
- volumeCounts5D(int16) - o mesmo que acima, mas para 5 dias
- volumeCounts7D(int16) - o mesmo que acima, mas para 7 dias

4. Descrição da solução

O primeiro a ser realizado para se chegar a uma solução será realizar um processamento nos dados para que os mesmos sejam normalizados e com isso sejam removidos dados incorretos (faltantes), colocados em escala equivalente e outras coisas. Desta forma, teremos os dados prontos para serem trabalhados.

Como solução do problema, será proposta uma rede neural recorrente(RNN), chamada de LSTM (Long short-term memory) [Wikipedia 2018]. As redes LSTM são boas quando se deseja classificar, processar e realizar previsões baseadas em dados contínuos (como é o caso dos dados do mercado financeiro).

Ao final será gerado um arquivo com o seguinte formato: ”time,assetCode,confidenceValue”sendo que o valor confidenceValue será entre 1 e -1, com 1 sendo esperado um grande aumento no ativo e -1 uma grande queda, para os próximos 10 dias.

5. Métricas de avaliação

Para esta competição do Kaggle [Kaggle 2018], o proposto é que se preveja um valor de confiança que é multiplicado pelo retorno ajustado do mercado financeiro de um determinado ativo em uma janela de 10 dias. Para cada dia no período de avaliação calcula-se:

$$x_t = \sum_i \hat{y_{ti}} r_{ti} u_{ti},$$

onde r_{ti} é o retorno inicial de dez dias ajustado pelo mercado para o dia t para o instrumento i e u_{ti} é uma variável do universo 0/1 (consulte a descrição dos dados para obter detalhes) que controla se um determinado ativo está incluído na pontuação em um determinado dia .

A pontuação é calculada como a média dividida pelo desvio padrão de seus valores x_t diários.

$$score = \bar{x_t}/\sigma\left(x_t\right)$$

Essa métrica de avaliação funciona como um *Sharpe ratio* que basicamente calcula a média por unidade, sendo que neste caso ele calcula a performance do portfólio, ou seja a média dividida pela volatilidade (std).

6. Análise do Problema

A exploração dos dados é divida conforme os dados, sendo que inicialmente é realizada uma exploração nos dados de mercado e em seguida uma exploração nos dados de notícias. Após realizada a exploração inicial é realizada uma análise exploratória também nesta ordem.

6.1. Exploração dos dados

Para começar é verificado que existem mais de 4 milhões de registros (linhas) nos dados de mercado, com 3511 ativos listados. Em seguida é verificada a coluna `returnsOpenNextMktres10` como demonstrado na figura abaixo, que apresenta que existem pontos fora da curva (outliers) e a maior parte dos dados está entre -0.2 e 0.2.

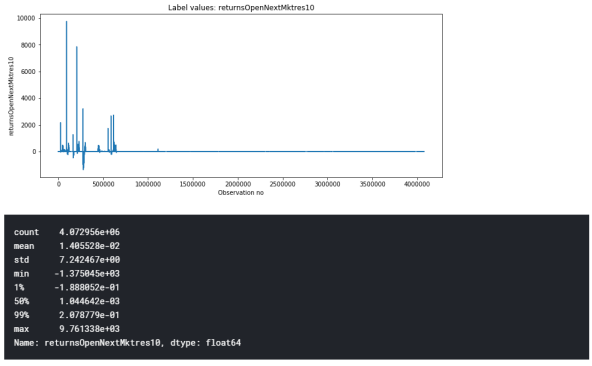

Figura 1. Retorno-rezidualizado do mercado no período de 10 dias para cada registro.

Para visualizar melhor os dados em questão foram escolhidos aleatoriamente um ativo para apresentar o preço, a volatilidade e o volume de suas negociações.

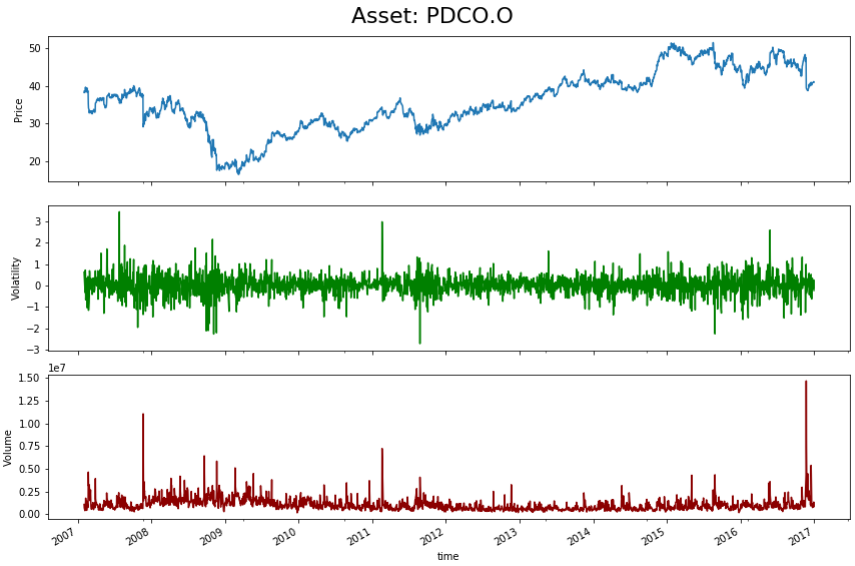

Figura 2. Preço, volatilidade e volume de negociações de um ativo aleatório nos dados.

Já os dados referentes as noticias apresentam mais de 9 milhões de registros (linhas) com noticias sobre 9677 ativos. Para entender um pouco melhor, foi criada a figura abaixo que representa qual atitude prevalece nas notícias, se mantém neutra, se tem uma atitude positiva ou negativa.

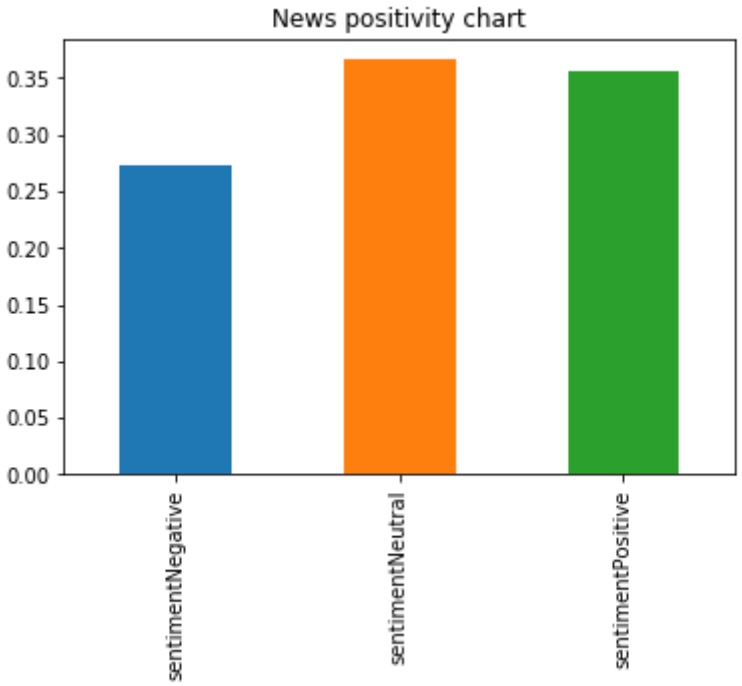

Figura 3. Gráfico que representa a positividade das notícias.

Como podemos verificar na figura acima, atitude neutra ou positiva são um pouco maior, de acordo com as notícias o mercado se move neutro com algumas pequenas tendencias.

6.2. Algoritmos e técnicas

Para o treinamento da rede neural, será utilizado a princípio um rede neural recorrente. As redes neurais recorrentes são redes que possuem conexões ponderadas dentro de uma camada [Christopher Olah 2015]. Ou seja, diferente das redes de feed-forward tradicionais em que a informação flui apenas para frente, as redes recorrentes possuem loops entre si (Figura 4). Podendo armazenar informações ao processar novas entradas, tornando ideal para séries temporais em que as entradas anteriores devem ser consideradas.

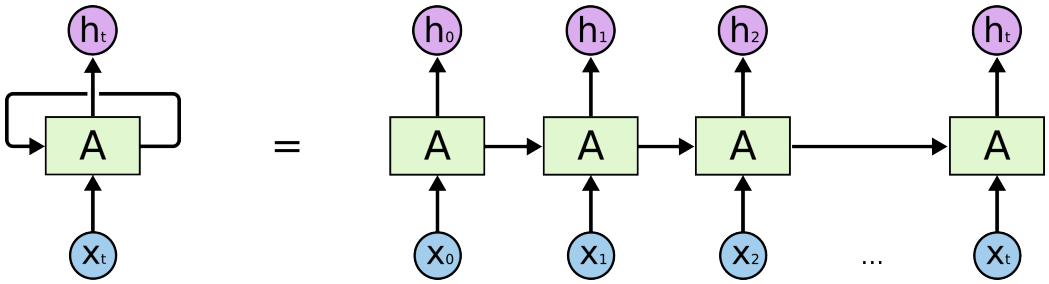

Figura 4. An unrolled recurrent neural network.

Para este problema também poderiam ser utilizados inumeras arquiteturas de redes recorrentes, porém foi optada uma rede recorrente bem conhecida chamada de LSTM (Long Short Term Memory) que é um tipo especial das redes neurais recorrentes. Elas foram introduzidas por Hochreiter Schmidhuber (1997) e foram melhoradas com o tempo por diversos autores, sendo utilizadas em uma grande variedade de problemas.

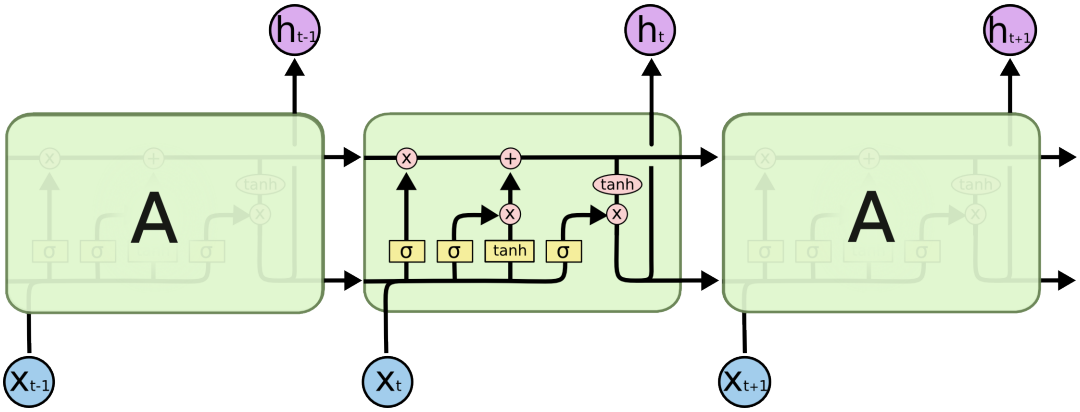

Figura 5. The repeating module in an LSTM contains four interacting layers.

A grande vantagem de se utilizar as LSTM‘s é que essas redes possuem a capacidade de adicionar ou remover informações nas celulas de estado, de forma bem regulada pelas estruturas chamadas de portões. Sendo que as LSTM possuem três destes portões para controlar as células de estado.

6.3. Modelo de referência

Por ser um problema específico, com dados específicos, o melhor modelo de referência será um trabalho disponível desta mesma competição, intitulado de ”Tuning hyper-params in LGBM achieve 0.66 in LB”.

Porém, por ser uma competição, a referência utilizada será o ranking disponibilizado pelo Kaggle [Kaggle 2018]. Ou seja, o resultado será comparado com os melhores resultados obtidos na competição até o momento de entrega do projeto final. Sendo que neste momento, a melhor pontuação é 5.65750.

7. Metodologia

Quanto a metodologia, primeiramente foi realizado um pré-processamento nos dados de forma a excluir os outliers, normalizar as colunas numéricas e transformar em números as colunas categóricas. Além disso, foram excluídas algumas colunas que não terão impacto no resultado. Já nos dados das notícias, como existem várias notícias durante o mesmo dia, para pré-processamento foi realizado uma agregação dos dados para melhorar a utilização do algoritmo. Na implementação da rede neural utilizando as LSTM, foi criado um novo dataset unificando as notícias e os dados de mercado para que o algoritmo funcionasse com ambas as informações.

7.1. Pré-processamento dos dados

Um gerador que gera dados lote por lote foi utilizado para alimentar o modelo LSTM. Para isso, os dados de mercado e noticias são pré-processados separadamente e então são fundidos para alimentar o modelo através da função *”fit_generate”*. Outro pré-processamento realizado nos dados de mercado foi a utilização somente dos dados após 2009, pois foi verificado que todos os ativos apresentam dados completos a partir de 2009.

Além disso, foram retirados os ruins dos dados de treinamento, incluídos dados faltantes e consertados alguns pontos fora da curva para normalizar os dados. As características categóricas nos dados foram transformadas em numéricas e por fim foram extraídos os dados de tempo, tudo isso apenas nos dados de mercado.

Nos dados de notícia, o processamento ocorreu normalizando valores numéricos, preenchendo dados faltantes e preparando os dados para serem ajuntados com os dados de mercado.

7.2. Implementação

Para a implementação primeiramente foi realizado a divisão dos dados entre dados de teste, dados de validação e dados de treinamento.

Figura 6. Separação dos dados entre treinamento, teste e validação.

Após separados os dados em treinamento, validação e teste, realizado o pre-processamento e criado o gerador para gerar os dados em lotes para alimentar o modelo, foi criado o modelo LSTM com três camadas, sendo a primeira camada com 128 unidades, a segunda camada com 64 unidade e a ultima camada com 32 unidades. Na camada de saída (*output*) foi inserida uma rede densa com função de ativação sigmoid.

7.3. Refinamento

Para realizar a calibração do modelo escolhido, escolhendo os melhores hiper-parâmetros para obter o melhor resultado, serão utilizados diversos parâmetros que são importantes nas LSTM-RNN, como taxa de aprendizado, taxa de abandono, ”epochs”e número de neurônios.

Uma das formas mais conhecidas e utilizadas para realizar a calibração é a pesquisa de grade [Jason Brownlee 2016]. Nela você insere como argumento um mapeamento do nome do parâmetro e uma lista de possíveis valores para serem testados. Estes testes são com base na acurácia das respostas, porém pode-se definir uma nova forma de calcular as pontuações.

Essas calibrações normalmente são bem demoradas por se tratar de um sistema de tentativa e erro, assim como o problema define claramente um limite de tempo e memória de execução, será optado pela utilização de apenas uma parte dos dados ao invés de utilizar todo o conjunto de dados disponíveis.

Esta etapa do problema foi bastante afetada pelas limitações impostas como memória máxima e tempo máximo, fazendo com que o refinamento não pudesse acontecer como esperado. Tiveram que ser utilizados poucos parâmetros nesta etapa para o refinamento, uma vez que todos os parâmetros consumiam muita memória e tempo. Assim, foram escolhidos o número de *epochs*, *batch_size*, *look_back* e *look_back_step* para realizar o refinamento, por acreditar que seriam relevantes na melhora do resultado, porém o ideal seria realizar todos os testes no futuro e verificar as melhores opções.

Antes do refinamento o resultado do modelo era inferior ao encontrado após o refinamento dos hiper-parâmetros. Porém, mesmo com o refinamento não foi possível obter resultados promissores como era esperado com este experimento. Sendo que uma possível solução seria realizar um refinamento melhor no futuro para tentar alcançar novos patamares de resultados.

8. Resultados

Apesar de diversas tentativas de encontrar melhores resultados, o modelo sempre apresenta uma baixa pontuação nas métricas definidas. É possível que a maneira que foi escolhida para agrupar os dados não foi boa, sendo que a otimização dos parâmetros ajudou mas não resolveu o problema da pontuação baixa.

Como pode ser observado na figura 7 abaixo, é possível perceber que durante o treinamento do modelo, o erro variou nas proximidades do 0.695 sendo que houve uma possível melhora com o tempo, porém insignificante. Já o erro de validação apresentou seu menor valor no inicio do treinamento, como ele determina a parada precoce caso não haja melhora em 5 passos, a rede parou o treinamento já no 6 passo.

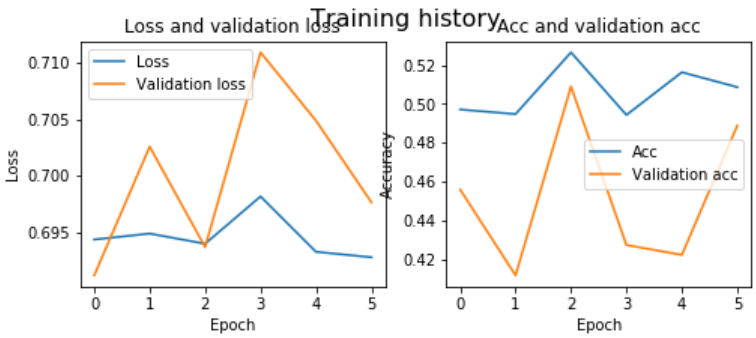

Figura 7. Histórico da perda (*loss*) a cada *epoch* no treinamento.

Assim, o valor da pontuação sigma obtido foi de 0.376 o que foi insuficiente para alcançar um resultado significante para solução do problema. Este resultado baixo acredita-se não ser devido ao modelo escolhido que é um modelo adequado ao problema, mas sim da forma com que levamos os dados em consideração para podermos respeitar as regras da competição. Caso fosse possível um treinamento mais longo poderíamos ter utilizado outros parâmetros que apesar de levarem mais tempo e recursos, levariam também a um melhor resultado.

8.1. Avaliação e validação do modelo

Na figura 8, podemos verificar como o modelo se comportou utilizando os dados do próprio treinamento para realizar as previsões. Desta forma, como demonstrado na figura, os resultados são muito próximas dos resultados verdadeiros, com a exceção de alguns pontos fora da curva.

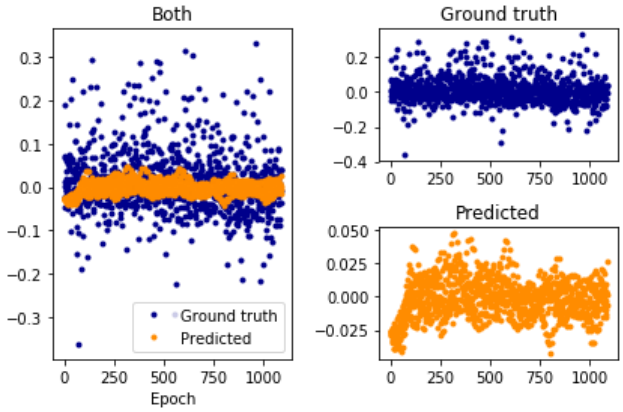

Figura 8. Representação da previsão do modelo utilizando os dados de treinamento.

Já na figura 9, podemos verificar como o modelo se comportou utilizando um ativo aleatório para realizar as previsões. Nele podemos perceber que as previsões não ficaram muito alinhadas com os resultados verdadeiros, sendo que isso se deve ao fato da baixa performance do modelo.

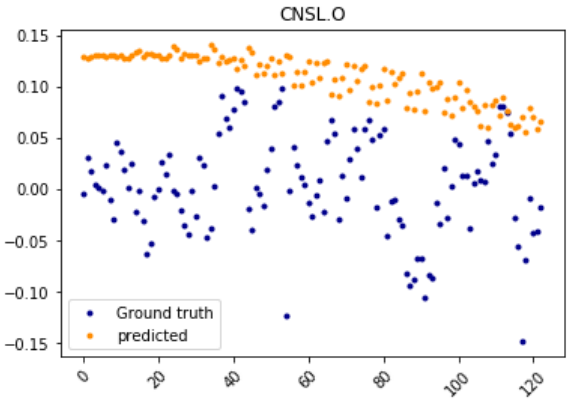

Figura 9. Representação da previsão em um ativo aleatório.

Com isso, é necessária uma reavaliação na forma como é tratada a geração de dados para alimentar o modelo, pois como foi feito não foi possível alcançar o objetivo de conseguir uma boa pontuação na competição.

8.2. Justificativa

Portanto, como demonstrado acima não é possível considerar o modelo desenvolvido suficiente para resolver adequadamente o problema proposto. Sendo que o modelo escolhido como *benchmark* conseguiu produzir uma pontuação sigma de 0.66 e atualmente no ranking da competição no Kaggle [Kaggle 2018] os melhores colocados apresentam pontuação como 5.74485, sendo que a competição foi encerrada no inicio de Fevereiro de 2019 não aceitando novas submissões. Como a pontuação vinha através da utilização do modelo em dados não disponibilizados, não é possível descobrir a pontuação do nosso modelo na competição.

9. Conclusão

Apesar da baixa performance do modelo o projeto foi capaz de demonstrar a simplicidade de se criar uma rede profunda na linguagem python. Sendo que mesmo não apresentando uma boa performance na pontuação e, portanto, na competição. A construção do modelo é relativamente simples, sendo que a maior dificuldade encontrada no projeto foi devido as restrições da competição que fizeram com que os dados de entrada no modelo fossem tratados de forma especial, dificultando o processamento do modelo. Inclusive também afetando o refinamento que foi considerado o segundo maior desafio deste projeto.

Para uma futura implementação seria interessante tentar inserir mais camadas na rede, diminuindo o número de unidades, o que poderia encontrar novos padrões nos dados. Sendo que a escolha deste projeto de trabalhar com 4 camadas (sendo uma a camada de saída e três ocultas) foi puramente devido as restrições da competição, que poderiam ser melhor pensados em se tratando de um problema tão complexo para resolver como este.

Concluindo, este foi um projeto que visou a tentativa de prever o mercado financeiro utilizando tanto dados do mercado quanto noticias, infelizmente o projeto não obteve o resultado esperado, sendo que o modelo LSTM escolhido para solucionar o problema não deve ser culpado, pois é uma escolha adequada ao problema, mas devido as restrições impostas não houve a completa utilização das suas possibilidades.

Referências

Christopher Olah (2015). Understanding lstm networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Online: Acessado em 30 de Outubro de 2018].

Intrinio: Data Intelligence, on demand (2018). Data market provider. https://intrinio.com/. [Online: Acessado em 30 de Outubro de 2018].

Jason Brownlee (2016). How to grid search hyperparameters for deep learning models in python with keras. https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/. [Online: Acessado em 31 de Outubro de 2018].

Kaggle (2018). Two sigma: Using news to predict stock movements. https://www.kaggle.com/c/two-sigma-financial-news. [Online: Acessado em 30 de Outubro de 2018].

Thomson Reuters (2017). News data provider. https://www.thomsonreuters.com/en.html. [Online: Acessado em 30 de Outubro de 2018].

Wikipedia (2018). Long short-term memory. http://www.bioinf.jku.at/publications/older/2604.pdf. [Online: Acessado em 30 de Outubro de 2018].