UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROJETO DE GRADUAÇÃO

Gabriel Mouzella Silva

REDES NEURAIS PROFUNDAS NO DESENVOLVIMENTO DE ROBÔ TRADER PARA MERCADO DE FOREX

Vitória-ES

Novembro/2018

Gabriel Mouzella Silva

REDES NEURAIS PROFUNDAS NO DESENVOLVIMENTO DE ROBÔ TRADER PARA MERCADO DE FOREX

Parte manuscrita do Projeto de Graduação do aluno Gabriel Mouzella Silva, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Vitória-ES

Novembro/2018

Gabriel Mouzella Silva

REDES NEURAIS PROFUNDAS NO DESENVOLVIMENTO DE ROBÔ TRADER PARA MERCADO DE FOREX

Parte manuscrita do Projeto de Graduação do aluno Gabriel Mouzella Silva, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Aprovado em 30 de Novembro de 2018.

COMISSÃO EXAMINADORA:

Prof. Dr. Jorge Leonid Aching Samatelo
Universidade Federal do Espírito Santo
Orientador

Prof. Dr. Klaus Fabian Coco
Universidade Federal do Espírito Santo
Examinador

Msc. Rodrigo Silva Cosme
Stevens Institute of Technology
Examinador

Vitória-ES

Novembro/2018

Gabriel Mouzella Silva

REDES NEURAIS PROFUNDAS NO DESENVOLVIMENTO DE ROBÔ TRADER PARA MERCADO DE FOREX

Parte manuscrita do Projeto de Graduação do aluno Gabriel Mouzella Silva, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Aprovado em 30 de Novembro de 2018.

COMISSÃO EXAMINADORA:

Prof. Dr. Jorge Leonid Aching Samatelo
Universidade Federal do Espírito Santo
Orientador

Prof. Dr. Klaus Fabian Coco
Universidade Federal do Espírito Santo
Examinador

Msc. Rodrigo Silva Cosme
Stevens Institute of Technology
Examinador

Vitória-ES

Novembro/2018

A todos aqueles que me apoiaram e me acompanharam nessa árdua jornada.

AGRADECIMENTOS

Aos meus pais, por haverem trabalhado arduamente para dar-me a melhor educação possível e todo o suporte necessário durante essa árdua jornada, à minha noiva por apoiar-me em minhas decisões e apoair-me durante os momentos mais difíceis, aos meus amigos por tornarem esses anos de universidades mais leves e aos professores que me ajudaram a trilhar o caminho até aqui.

Ao professor Jorge Leonid Aching Samatelo por sua orientação e dedicação em ensinar e guiar-me, estando sempre auxiliando-me e mostrando os caminhos que seguir para tornar possível a realização deste trabalho.

À banca examinadora pela aceitação do convite e pelo tempo investido para leitura e avaliação desse trabalho.

RESUMO

Desde o século XVII com a Companhia Holandesa das Índias Ocidentais, primeira empresa de capital aberto, o homem vem desenvolvendo métodos de análises a fim de definir o melhor momento para comprar ou vender ações. Na década de 1970 com o advento do computador, a utilização de métodos lineares para tentar prever tendências, tais como médias móveis e ARIMA[1] entre outros, foi largamente utilizado, e ainda hoje tem seu espaço no mercado financeiro.

Porém séries temporais financeiras são bastantes desafiadoras com relação a previsões, considerando que são, muitas vezes, dependentes de fatores externos como notícias econômicas e políticas. Portanto poucos modelos encontrados representam de maneira fidedigna a movimentação de ativos, e aqueles que encontram modelos que o fazem não o tornam público.

A partir do aumento do poder computacional, modelos não-lineares passaram a ser estudados a fim de tentar prever o mercado financeiro, dentre esses modelos encontram-se as redes neurais, uma vez que são poderosas ferramentas na análise de séries temporais. Portanto esse trabalho propõe a utilização de redes neurais recursivas, do tipo LSTM, a fim de tentar prever a movimentação de ativos do mercado de ForEx, para tanto será discutido, no decorrer desse projeto o conceito de redes neurais recursívas LSTM, assim como as arquiteturas que apresentaram melhor desempenho.

Palavras-chave: *Deep-learning*; *Long Short Term Memory*; *Foreign Exchange*.

[1] *Autoregressive Moving Average*

ABSTRACT

Since the XVII century with the creation of the Dutch East India Company, the first open capital company, men developed methods of analisys in order to define the best moment to buy or sell stocks. In the 1970s due to the appearance of the first computers, linear methods were largely used to try to predict trends (such as moving averages and ARIMA between others) and still have their space on today's market.

However, financial time series are challenging to predict considering that sometimes it depends on external factors such as economic and political news. Therefore, few models represent well the stocks movements and those who find these models do not make them public.

Since the rise of computational power, non-linear models started to be studied as a way to try to forecast the financial market. Between those models are neural networks, which are a poweful tool for time series analisys. Hence, this project proposes the use o LSTM recurrent neural network as a way to try to forecast the movements of stocks on the ForEx market. On this work, the concept of LSTM recurrent neural network and the models that presented the best performance will be discussed.

Keywords: Deep-learning; Long Short Term Memory; Foreign Exchange.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CNN	*Convolutional Neural Network*
DL	*Deep Learning*
GPU	*Graphics Processing Unit*
LSTM	*Long Short Term Memory*
ML	*Machine Learning*
MQL5	*MetaQuotes Language 5*
SGD	*Stochastic Gradient Descent*
UFES	Universidade Federal do Espírito Santo

SUMÁRIO

1 INTRODUÇÃO

1.1 Apresentação e Objeto de Pesquisa

Segundo a pesquisa trienal feita em 2016 pelo Banco de Compensações Internacionais (*Bank for International Settlements*) o mercado de ForEx [1] movimentou em média $5,1$ trilhões de dólares por dia em Abril de 2016 (SETTLEMENTS, 2016), o equivalente a $2,9$ vezes o PIB Brasileiro em 2016, que foi de R$ $6,259$ trilhões (U$ 1,79 trilhões) (BRASIL, 2018b). Portanto o mercado de câmbio é muito líquido, possibilitando que ordens sejam efetuadas sem impactar na variação do preço do ativo.

O aumento da tecnologia nas últimas décadas e a democratização da internet possibilitou a digitalização das operações de compra e venda de ativos, tornando possível tornar-se sócios de empresas, pela compra de ações, a partir do computador de casa ou até mesmo do celular. Consequentemente um dos campos da automação que se desenvolveu exponencialmente foi primeiramente o da automatização de ordens e em seguida o de robôs *trades*, muitas vezes operando a partir de análises técnicas, comumente empregadas por analistas financeiros. Porém com o aumento do poder de processamento dos computadores e o grande avanço das pesquisas no campo *Machine Learning* - ML (tradução livre, aprendizado de maquinas), observou-se um aumento do interesse em algoritimos que façam uso de modelos de aprendizagem na busca de melhores retornos dos investimentos ao mesmo tempo em que se reduz o risco.

O mercado de capitais é um mercado global, diferenciando-se majoritariamente em legislatura. Portanto o desenvolvimento de conhecimento na área possibilita a inserção no mercado de trabalho em qualqer país. Pode-se considerar também que devido ao aumento gradual da idade de aposentadoria e a redução do teto do INSS (Instituto Nacional do Seguro Social) para as novas e futuras gerações, faz-se necessário o acumulo de capital financeiro de modo que produza uma renda passiva como alternativa para aposentar-se. Porém a prática de investir demanda tempo e estudo. Portanto a utilização de um algoritimo que faça tais tarefas vem a ser benéfica, uma vez que possibilita ao investidor deliberar tempo em um trabalho que lhe gere renda ativa.

Desde o princípio do mercado de capitais, o homem vem tentando prever movimentações financeiras como forma de manter-se à frente dos demais e obter maiores retornos financeiros.

[1] O mercado de câmbio é o ambiente onde se realizam as operações de câmbio entre os agentes autorizados pelo Banco Central e entre estes e seus clientes. O mercado de câmbio brasileiro é regulamentado e fiscalizado pelo Banco Central e compreende as operações de compra e de venda de moeda estrangeira (BRASIL, 2018a). O mercado de câmbio é comumente referido como ForEx, abreviação para *Foreing Exchange*.

Com o advento da computação os cálculos de previsões foram ficando mais fáceis e dados passaram a ser coletados e guardados para análise futura. Tais dados históricos financeiros, muitos hoje à disposição dos *traders*, podem ser estudados de forma a entendê-los e modelá-los. Graças à grande quantidade de dados disponíveis os sistemas estatísticos e de aprendizagem, têm conseguido com boa acurácia, modelar e prever dados, trazendo a seus operadores bons retornos financeiros.

Na atualidade, existe uma área de pesquisa relacionada a ML denominada *Deep Learning* - DL (tradução livre, aprendizado profundo) que permite implementar um modelo de analise de dados *end-to-end* (GOODFELLOW; BENGIO; COURVILLE, 2016b), ou seja, máquinas de aprendizagem que têm a capacidade de extrair características de importância dos dados e simultaneamente efetuar a tarefa de classificação ou regressão com alto nível de precisão. Nesse contexto, as redes neurais estão superando a performance humana em muitas áreas onde são aplicadas, seja a tarefa uma classificação de imagens (HE et al., 2015) ou jogos altamente subjetivos (OPENAI, 2017) (SILVER et al., 2016) até o gerenciamento de portfólios financeiros e negociações de ativos (*trading*) (STOPPIGLIA; IDAN; DREYFUS, 1998).

Entre as arquiteturas de interesse para analise de series temporais tem-se a Rede Neural Recorrente (RNN) e a rede *Long Short Term Memory* (LSTM). Uma rede RNN tem como característica usar conexões de *feedback* para estocar informações em memória de curto prazo. Porém tardam muito tempo em aprender que informação estocar ou então não funcionam, seja devido ao problema de *vanishing gradient* (tradução livre, desvanecimento do gradiente) ou ao problema de *exploding gradient* (tradução livre, explosão do gradiente) (HOCHREITER; SCHMIDHUBER, 1997). Diferentemente, uma rede LSTM permite contornar os problemas de *vanishing gradient* e *exploding gradient* tornando possível a criação de modelos recurrentes com varias camadas com a capacidade de aprender características de longo praço (GREFF et al., 2017).

Tomando em conta o exposto, neste trabalho, é proposto desenvolver uma rede LSTM que se demonstre adaptável a diferentes ativos do mercado de câmbio, considerando que, o problema principal a enfrentar é a imprevisibilidade do mercado, que pode variar em qualquer direção e não, necessariamente, segue uma tendência. Após o desenvolvimento da arquitetura, os únicos pré-requisitos são um banco de dados suficientemente grande (quantidade de experiência), poder de processamento computacional (velocidade de aprendizado) adequado e boas ferramentas de análises de resultados.

1.2 Trabalhos Relacionados

Dentre os diversos artigos lidos para a elaboração deste trabalho pode-se citar como de importância especial:

Em (NELSON; PEREIRA; OLIVEIRA, 2017) os autores fazem uso de uma rede LSTM para prever se o valor do ativo aumenta ou não em um intervalo de 15 minutos, sem levar em consideração o percentual da variação. Compara-se os resultados com os modelos *Random Forest* e *Multi-Layer Perceptron*, assim como com as técnicas de investimentos *Buy and Hold*[2]. O conjunto de dados foram coletados de ativos pertencentes ao índice Bovespa no período de 2008 a 2015 com período de tempo de 15 minutos. Conclui-se que as redes LSTM superam, na maioria das vezes, outras abordagens.

Em (HANSSON, 2017) O autor utiliza uma rede LSTM para predição da direção de movimento do mercado em intervalo diário dos índices Bovespa 50, S&P 500 e OMX 30. Os resultados são comparados com diferentes modelos, dentre eles o modelo ARMA (*Autoregressive Moving Average*), comumente utilizado no mercado financeiro. Os resultados indicam que uma rede LSTM pode ser utilizada a fim de prever séries temporais financeiras. Aparenta-se que é possível prever a direção da bolsa de valores Suéca, porém o mesmo não se aplica às bolsas de valores Estado Unidense e Brasileira, o que sugere que as bolsas Estado Unidense e Brasileiras são mais impulsionados por dados econômicos e notícias quando comparada com o mercado Suéco.

Em (NASCIMENTO, 2018) o autor avalia a viabilidade do uso de redes neurais *Multilayer Perceptron* (MLP) e redes neurais sem peso VG-RAM como preditores para o mercado futuro brasileiro na escala de dezenas de segundos. Os resultados indicaram que o mercado futuro brasileiro é um mercado eficiente[3] na maior parte dos dias e portanto se comporta como uma série *random walk*, o que justifica a dificuldade em se fazer previsões com boa acurácia.

Em geral, os trabalhos citados têm as seguintes restrições: usam técnicas de classificação direcional de mercado, ou seja buscam prever somente se o mercado irá subir ou cair, sem levar em consideração a variação absoluta ou percentual do valor do ativo, e se restringem

[2] *Buy and Hold* é uma estratégia de investimento em que o investidor compra o ativo e o detém por um longo período de tempo, independente de flutuações do mercado. Um investidor que utiliza a estratégia de *Buy and Hold* seleciona os ativos porém não se preocupa com variações de preços no curto prazo e nem com indicadores técnicos.(INVESTOPEDIA, 2018)

[3] Mercado eficiente: estado em que toda informação é completa e imediatamente refletida no preço de um ativo, portanto um investidor não deve esperar retornos anormais (acima do mercado) a partir de análise técnica ou fundamentalista(NASDAQ, 2018)

à predição de somente um horizonte de tempo.

Tomando em conta os artigos citados e as restrições que apresentam, a proposta deste trabalho está baseada no uso de uma rede LSTM para predizer variações absolutas do ativo em diferentes horizontes de tempos. Os algoritimos apresentados são extremamente especializados em seus horizontes de predição e ativos, a fim de melhorar seus desempenhos. Portanto o mesmo só apresenta resultados satisfatórios quando executado no gráfico com o período de tempo e ativo específicos, e para qualquer outra configuração os modelos não apresentam resultados consideráveis. Por conseguinte, um dos problemas dessa aproximação é a necessidade de treinar tantos modelos quanto ativos e períodos de tempos se queira operar.

1.3 Objetivos

Objetivo Geral

- Implementação de uma rede neural recorrente LSTM que seja capaz de fazer predições de valores de ativos do mercado de Forex.

Objetivos Específicos

- Estudar a arquitetura de uma rede neural recorrente LSTM e como aplicá-la;
- Implementar uma rede neural LSTM capaz de prever movimentações nos preços de ativos do mercado de câmbio;
- Validar e testar a rede neural LSTM junto a um *Expert Advisor* no programa *Home broker Meta Trader*;

2 EMBASAMENTO TEÓRICO

2.1 Introdução

Este capítulo tem por finalidade estabelecer os conceitos teóricos necessários usados no trabalho. Portanto, o capítulo inicia-se com a definição de séries temporais, após é descrito o funcionamento de Rede Neural e de Redes Neurais Profundas e como elas são caracterizadas, em continuação é descrito o funcionamento de uma rede LSTM, começando por expor suas vantagens e logo uma exposição de sua estrutura.

2.2 Séries temporais

Dados obtidos de observações sequenciais no decorrer do tempo são extremamente comuns. A lista de áreas onde séries temporais são estudados é ampla, podendo citar dentre algumas: finanças, meteorologia, agricultura, ciências biológicas e ecologia (MAINDONALD, 2009).

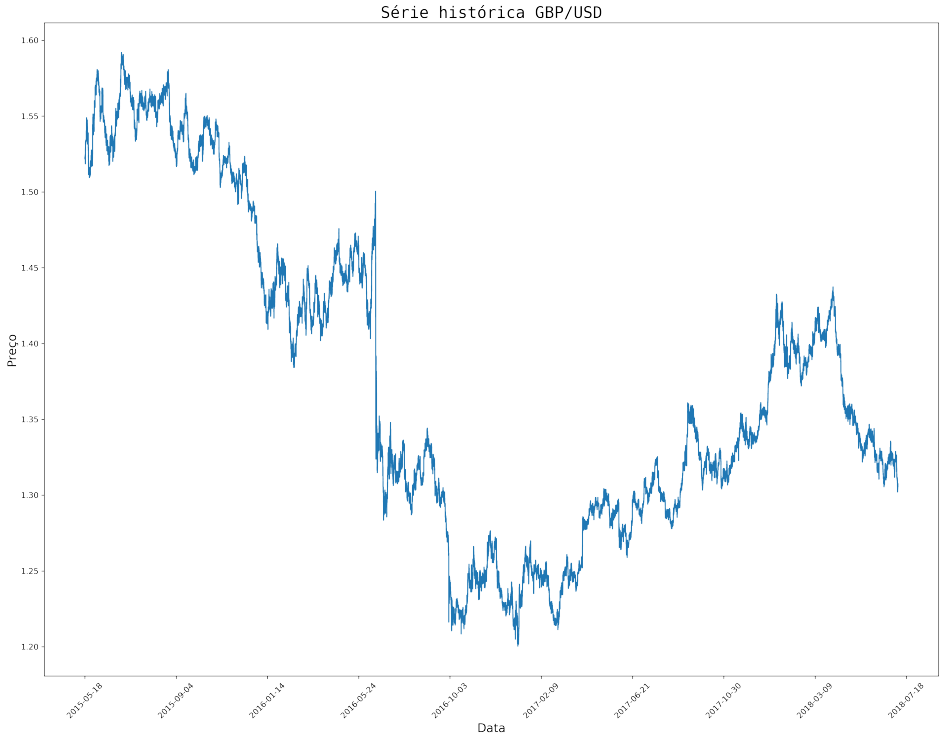
O propósito da análise de séries temporais geralmente divide-se em: entender ou modelar a representação estocástica de uma determinada série e predizer o valor futuro da série baseado no histórico de dados (MAINDONALD, 2009).

Um exemplo de séries temporais é apresentado abaixo na Figura 1, que representa a variação do ativo GBPUSD de meados de 2016 a meados de 2018. Sendo esta a série utilizada para o treinamento, validação e teste das redes neurais aqui apresentadas.

Séries temporais financeiras são compostas por dados como: data e hora, abertura, máxima, mínima, fechamento, volume e *spread*[1]. Para o desenvolvimento das redes neurais aqui apresentadas serão utilizadas somente os dados de: abertura, máxima, mínima e fechamento, conhecido por OHLC (*Open, High, Low, Close*). É possível observar através da Figura 1 que séries financeiras são bastante ruidosas, devida à alta volatilidade no preço do ativo, e algumas vezes consideradas randômicas, tornando-se um problema desafiante fazer predições da mesma.

[1] É a diferença entre o preço atual de compra e de venda para um determinado ativo (*spread* de compra/venda)(DEFINITION..., 2018).

Figura 1 – Série temporal financeira do ativo GBP/USD com 80.000 pontos

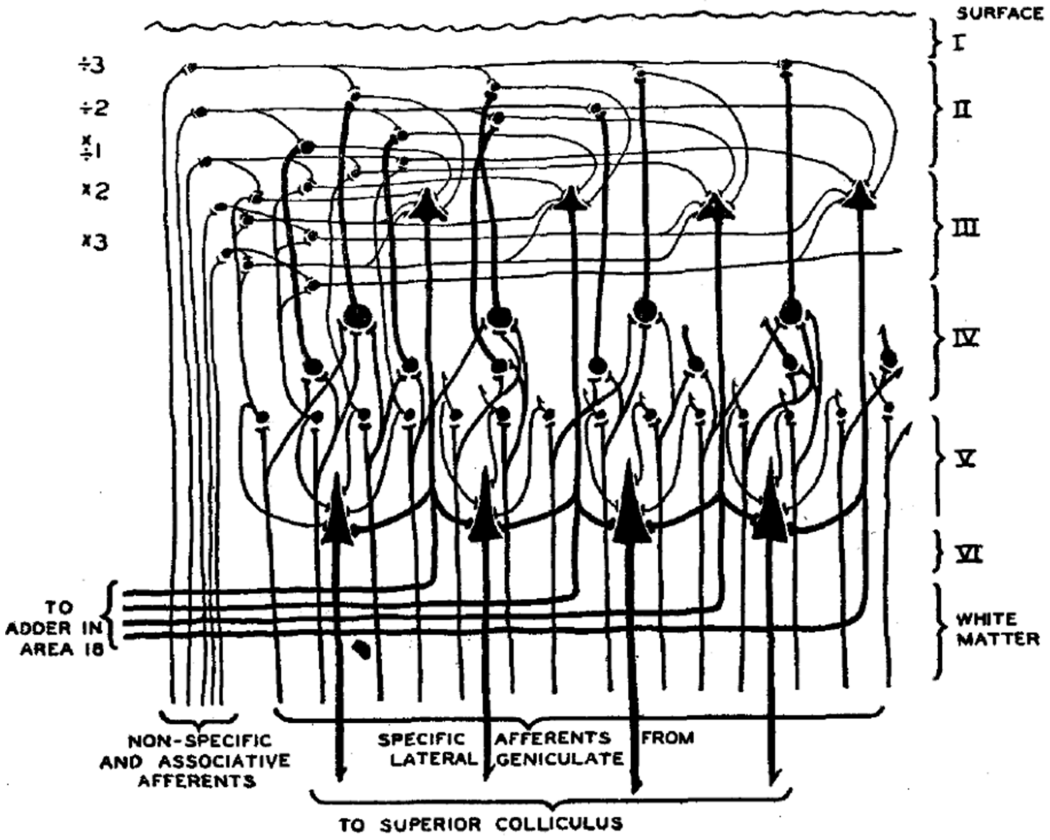


2.3 Redes Neurais

O conceito de Redes Neurais surgiu na década de 1940 com o desenvolvimento de teorias sobre o aprendizado biológico com o modelo de neurônio de McCulloch-Pitts, que compunha um modelo inicial das funções do cérebro (GOODFELLOW; BENGIO; COURVILLE, 2016a). Na Figura 2 é representado parte dos neurônios em uma seção vertical do cortex tirada radialmente (PITTS; MCCULLOCH, 1947), que ilustra sua teoria de como somos capazes de perceber objetos em diferentes localizações na retina e com diferentes tamanhos e posições, chamadas invariâncias. O motivo principal é a repetição de padrões das conexões neurais no cortex, principal característica das Redes Neurais Convulocionais (SEJNOWSKI, 2018). Tal modelo pode reconhecer duas categorias diferentes de entradas testando se a mesma era positiva ou negativa (GOODFELLOW; BENGIO; COURVILLE, 2016a).

Em (ROSENBLATT, 1958) é introduzido o conceito de *perceptron*, que consiste me um algoritimo de classificação binária. A integralização das entradas são implementadas a partir da adição dos pesos obtidos durante o treinamento, caso o resultado da adição seja maior que um limiar estabelecido o neurônio implementa 1 como saída, caso contrário 0.

Figura 2 – Representação de parte dos neurônios em uma seção vertical do cortex. Fonte: (PITTS; MCCULLOCH, 1947)



Rosenblatt conclui, dentre outros, que:

- Em um ambiente de estímulos randômicos, um sistema que consista em unidades randomicamente conectadas pode aprender a associar respostas específicas a estímulos específicos. Mesmo que muitos estímulos estejam associados a cada resposta, eles ainda podem ser classificados.
- Em um ambiente ideal a probabilidade de uma resposta correta é inversamente proporcional ao número de estímulos aprendidos.
- Séries temporais de padrões de estímulos e de respostas podem ser aprendidos por um sistema que use apenas uma parte dos princípios originais de separabilidade estatística, sem introduzir nenhum complicador na organização do sistema.
- A memória do *perceptron* é distribuída, no sentido de que qualquer associação pode fazer uso de uma grande parte das neurônios do sistemas, e a remoção de uma porção da associação do sistema não teria um efeito apreciável de nenhuma associação do classificação em particular, mas apareceria como um defeito generalizado em todas as associações aprendidas.

O *perceptron* foi o primeiro modelo capaz de aprender os pesos dado exemplos de diferentes categorias. O *Adaptative Linear Element* - ADALINE (tradução livre, elemento linear adaptativo), que data da mesma época, retorna o valor da função em si para predizer um número real, e podia também aprender a predizer tais números a partir dos dados (GOODFELLOW; BENGIO; COURVILLE, 2016a).

2.4 Redes Neurais Profundas

Uma Rede Neural padrão consiste de neurônios conectados com uma camada de entrada, uma única camada escondida e uma camada de saída, onde cada um produz uma sequência de valores reais de ativação. Neurônios da camada de entrada são ativados a partir de sensores de percepção do meio, outros neurônios são ativados a partir de pesos das conexões de neurônios de camadas anteriores (SCHMIDHUBER, 2015). Busca-se com o aprendizado que a Rede neural apresente um comportamento desejado, como por exemplo previsão de preços de ativos no mercado de câmbio. Dependendo da complexidade do problema e de como estão organizados os neurônios, tal comportamento requer mais de uma camada escondida, onde cada camada transforma (normalmente transformações não-lineares) os pesos da rede (SCHMIDHUBER, 2015), tal configuração com duas ou mais camadas escondidas é conhecida como Redes Neurais Profundas (RNP).

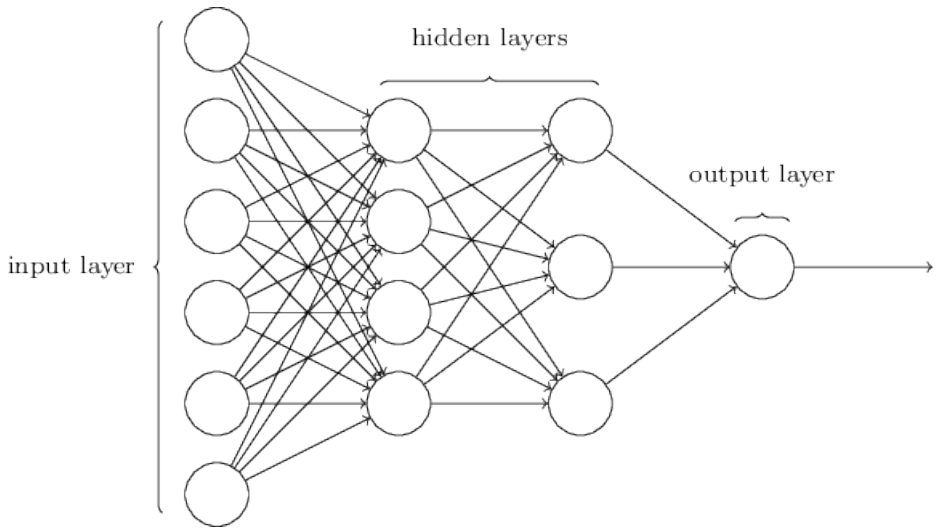


Figura 3 – Exemplo de uma Rede Neural Profunda com duas camadas escondidas. Fonte: (GOODFELLOW; BENGIO; COURVILLE, 2016a)

2.5 RNN

As redes RNN são modelos de conexões com a habilidade de seleção de passagem de informação por uma sequência de passos enquanto processa dados sequenciais um elemento por vez. Ademas, redes neurais recorrentes pode simultaneamente modelar séries temporais em múltiplas escalas (Lipton; Berkowitz; Elkan, 2015). RNN são redes neurais Redes neurais profundas com a saída da camada oculta realimentada, introduzindo assim a noção de tempo no modelo (Lipton; Berkowitz; Elkan, 2015).

2.6 redes LSTM

As redes LSTM foram propostas em (HOCHREITER; SCHMIDHUBER, 1997), e tinham como objetivo resolver o problema de *vanishing gradient*. Tal modelo se assemelha a uma RNN padrão com uma camada escondida, porém cada nó é substituido por uma célula de memória, o que assegura que o gradiente passe por muitas etapas sem desaparecer ou explodir(Lipton; Berkowitz; Elkan, 2015).

Para construir uma arquitetura que permita o fluxo de erro constante sem as desvantagens da RNN (desaparecimento e explosão de gradiente), introduziu-se características adicionais às células. Um *gate* de entrada multiplicativo foi introduzido de forma a evitar perturbações de entradas irrelevantes advindas de células anteriores. Da mesma maneira introduziu-se um *gate* de saída para proteger as células seguintes de perturbações de conteúdos estocados na memória atual (HOCHREITER; SCHMIDHUBER, 1997).

O resultado é uma célula de memória constituída por uma unidade central linear com *Constant Error Carousel* (CEC) fixada, onde cada unidade contém informações relevantes sobre o estado atual da rede. Um *gate* de entrada pode usar entradas de outras células de memória a fim de decidir se guardar ou esquece determinada informação em sua célula.

O *gate* de entrada terá que aprender quando abrir mão dos erros, escalando-os apropriadamente. Basicamente os *gates* multiplicativos abrem e fecham acessos ao fluxo de erro constante CEC (HOCHREITER; SCHMIDHUBER, 1997).

O funcionamento de uma célula de memória é tal que, primeiramente, deve-se decidir que informação deve ser esquecida, tal decisão é feita por uma função sigmoidal chamada "Função de esquecimento", que ao analisar a entrada no momento x_t e a saída do estado anterior h_{t-1} tem como resultado um número entre 0 e 1, onde 1 represente manter toda a

informação e 0 esquecê-la completamente (OLAH, 2018).

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.1}$$

Em seguida deve-se decidir quais informações reter na célula. Tal processo é feito em duas partes, primeiramente a função de entrada sigmoidal decide quais valores atualizar. Depois é criado um vetor de valores candidatos $\vec{C_t}$, que poderiam ser adicionados ao estado (OLAH, 2018).

$$i_t = \sigma(W_i \cdot [h_{t-1,x_t}] + b_i) \tag{2.2}$$

$$\vec{C_t} = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2.3}$$

É feito então a atualização do estado da célula, C_{t-1} em seu novo estado C_t, onde é efetivamente feito o esquecimento daquilo que deve-se esquecer e a retenção de novos dados.

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \vec{C_t} \tag{2.4}$$

Por fim é decidido qual o valor de saída da célula, onde o valor de saída é baseado é baseado no estado da célula, que será o resultado de uma função sigmoidea multiplicado pela *tanh* do estado da célula C_t, assim somente os valores esperados aparecerão na saída.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.5}$$

$$h_t = o_t \cdot tanh(C_t) \tag{2.6}$$

Todo o funcionamento acima descrito é representado visualmente pela Figura **??** abaixo, que demonstra o funcionamento de uma célula LSTM e sua interação com demais células.

2.7 Resumo

Deste capítulo pode-se concluir que: A ideia do funcionamento de uma rede neural vem sendo estudado a 7 décadas, desde a introdução da rede Perceptron em 1947. A partir de

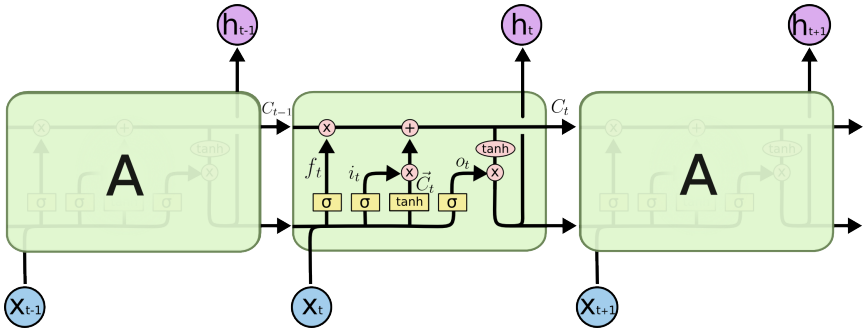


Figura 4 – Estrutura da célula de memória LSTM e sua conexão com outras camadas. Fonte: (OLAH, 2018), adaptado

então surgiu-se novas arquiteturas, que ajudou a formular o que hoje conhecemos como redes neurais, como é o caso da LSTM, porém foi somente com o avanço computacional nas últimas duas décadas que tal campo de estudos ganhou destaque. O que segue deste trabalho são implementações das partes constitutivas do sistema de predição desenvolvido. Portanto, os seguintes capítulos estão dedicados especificamente às etapas de processamento e predição.

3 PROPOSTA

3.1 Introdução

Este trabalho tem por objetivo a criação de um robô investidor, capaz de operar no mercado de câmbio internacional utilizando-se de técnicas de redes neurais profundas para maximizar o retorno do investimento reduzindo ao máximo a exposição ao risco. Buscar-se-á criar, neste trabalho, um algoritimo capaz de adaptar-se a aprender e operar o par de moedas apresentado a ele.

Para o desenvolvimento do trabalho decidiu-se por utilizar o par de moedas GBP/USD (*Great Britain Pound/US Dollar*, Libra Esterlina/EUA Dólar) por ser o terceiro par de moedas mais negociado no mercado de ForEx com $9,3\%$ do volume diário (SETTLEMENTS, 2016), por ser um ativo não tão volátil como o EUR/USD (Euro/EUA Dólar) e o JPY/USD (*Japanese Yene/US Dollar*, Yene Japonês/EUA Dólar), primeiro e segundo ativos mais negociados, respectivamente, e apresentar uma base de dados bastante completa.

Sendo assim, o capítulo divide-se em três seções além desta introdução. Na primeira seção, é feita explicado o processo de desenvolvimento da rede neural, desde a aquisição dos dados até a seleção da melhor arquitetura e o desenvolvimento de um algoritimo de retreinamento das redes LSTM quando necessário. Na segunda seção, são descritos os métodos de seleção das melhores redes neurais e apresentação de suas arquiteturas. Finalmente, na ultima seção é descrito o algoritimo de comunicação entre as plataformas Python e o *Meta Trader*, tornando possível a realização de operações a partir de uma corretora.

3.2 Técnica Proposta

Nesta seção é descrita a técnica proposta para predição de séries temporais baseada no uso de redes LSTM. Tal proposta está dividida em 4 etapas, especificamente:

- **Etapa de Captura dos dados**. A captura dos dados se deu a partir do captura *Meta Trader* utilizando-se da base de dados fornecida pela corretora Robo Forex, cujos dados incluidos são abertura, máxima, mínima e fechamento. Após a aquisição dos dados é feita a imputação de dados faltantes e por fim sua reorganização de

forma a estarem no formato requerido pela rede neural. Os dados foram divididos em 80% treino, 10% validação e 10% teste, sendo que o teste somente é feito ao final do treinamento.

- **Etapa de treinamento das redes LSTM**. As redes LSTM foram treinadas variando-se os parâmetros, buscando-se a melhor combinação de parâmetros para o ativo em questão. O algoritimo foi desenvolvido buscando-se a generalização de parâmetros como forma de abrangir qualquer combinação com mínimo esforço. Por fim cada uma das redes foi salva em um diretório e o resumo de seus parâmetros e métricas de desempenho salvos em um arquivo CSV para análise futura.

- **Etapa de seleção das redes LSTM com melhor desempenho**. Após os treinamentos das redes neurais, foi feita uma comparação entre os resultado utilizando como métrica de avaliação o MSE (*Mean Squared Error*) em relação ao conjunto de teste.

- **Etapa de retreinamento das redes LSTM selecionadas**. Por fim, os modelos selecionados passaram por uma etapa de retreino a fim de evitar problemas de *concept drift*[1] ao utilizar dados mais recentes.

As etapas de treinamento, seleção e retreinamento das redes LSTM foram projetadas de maneira a generalizar para qualquer base de dados advinda do *Meta Trader* e assim possibilitar o treinamento de redes neurais LSTM para qualquer que seja o ativo desejado. Decidiu-se por proceder assim como forma de facilitar a escalabilidade do programa, possibilitando sua utilização em larga escala. A seguir cada etapa será explicada em detalhe nas próximas subseções.

3.2.1 Etapa de Captura dos dados

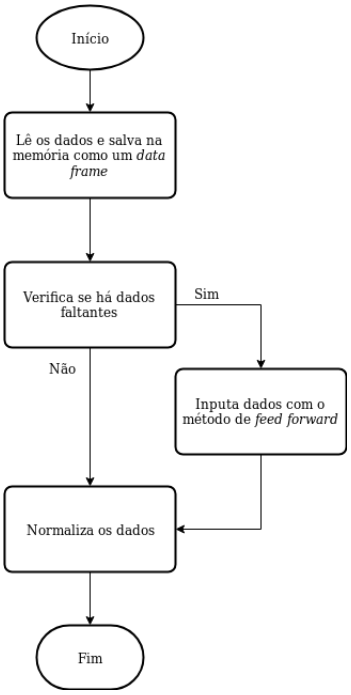
Para a captura da base de dados utilizou-se o programa *Meta Trader* conectado à corretora de ForEx *Robo Forex*[2], corretora de mercado de câmbio e metais preciosos sediada na Inglaterra, que apresenta extensa base de dados para diversos ativos.

Após a captura dos dados, faz-se necessário verificar a base de dados por inconsistências, e repará-la quando estas existem. Portanto desenvolveu-se um algoritimo de limpeza e imputação de dados, utilizando do método de *feed forward*[3] quando necessário. O

[1] *Concept drift* se refere à mudança de relação entre entrada e saída de dados em um problema proposto no tempo (BROWNLEE, 2018).
[2] https://roboforex.com
[3] Substitui o valor faltante pelo último valor observado

fluxograma abaixo explicita o funcionamento de tal algoritimo. Cabe indicar que os dados resultantes são valores normalizados e portanto devem ser desnormalizados (as equações que descrevem a normalização e desnormalização encontram-se nas Equações 3.1 e 3.2) ao final da predição para comparação comparação e utilização em casos reais.

Figura 5 – Diagrama de fluxo da Etapa de Captura dos dados (Fonte: próprio autor)



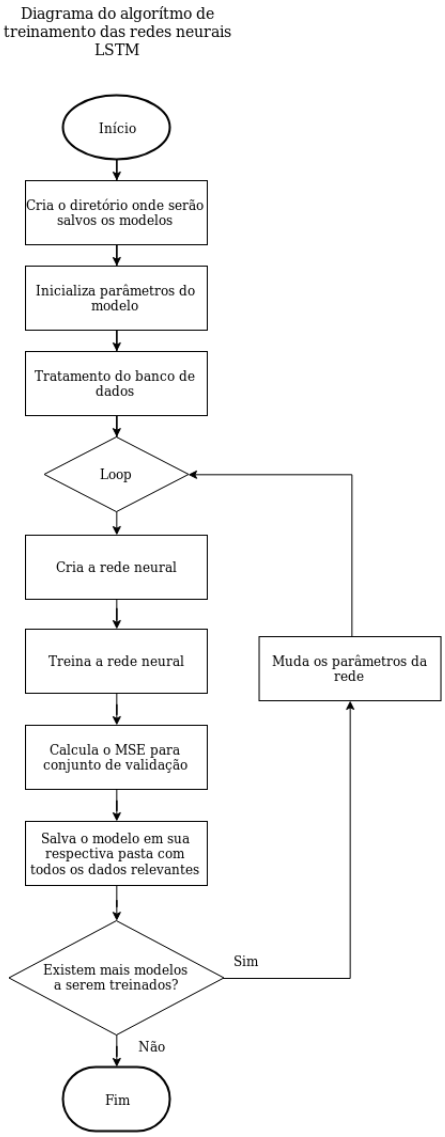
$$y_n = \frac{y - min}{max - min} \tag{3.1}$$

$$y = y_n \cdot (max - min) + min \tag{3.2}$$

3.2.2 Etapa de treinamento das redes LSTM

A descrição em diagrama de fluxo desta etapa é mostrada na Figura 6. O objetivo da mesma é treinar diversas redes neurais, mudando seus parâmetros a cada treino, guardar o modelo da rede junto com o histórico por época em um diretório e o resumo da rede em um arquivo CSV para análise futura. É importante salientar que todos as arquiteturas são inicialmente salvas, e as que obtiverem pior *performance* são excluídas durante o processo de seleção, visando economizar recurso de armazenamento.

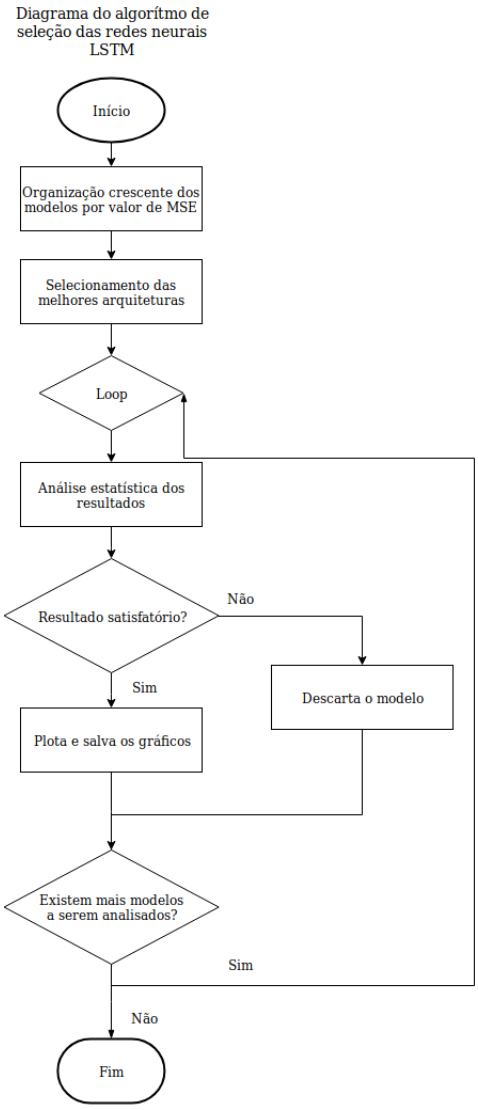
Figura 6 – Diagrama de fluxo da etapa de treinamento das redes LSTM (Fonte: próprio autor)



3.2.3 Etapa de seleção das redes LSTM com melhor desempenho

A descrição em diagrama de fluxo desta etapa é mostrada na Figura 7. O objetivo da mesma é selecionar os melhores modelos, gerando suas análises estatísticas e gráficos, e descartar os modelos cujo MSE não são satisfatórios.

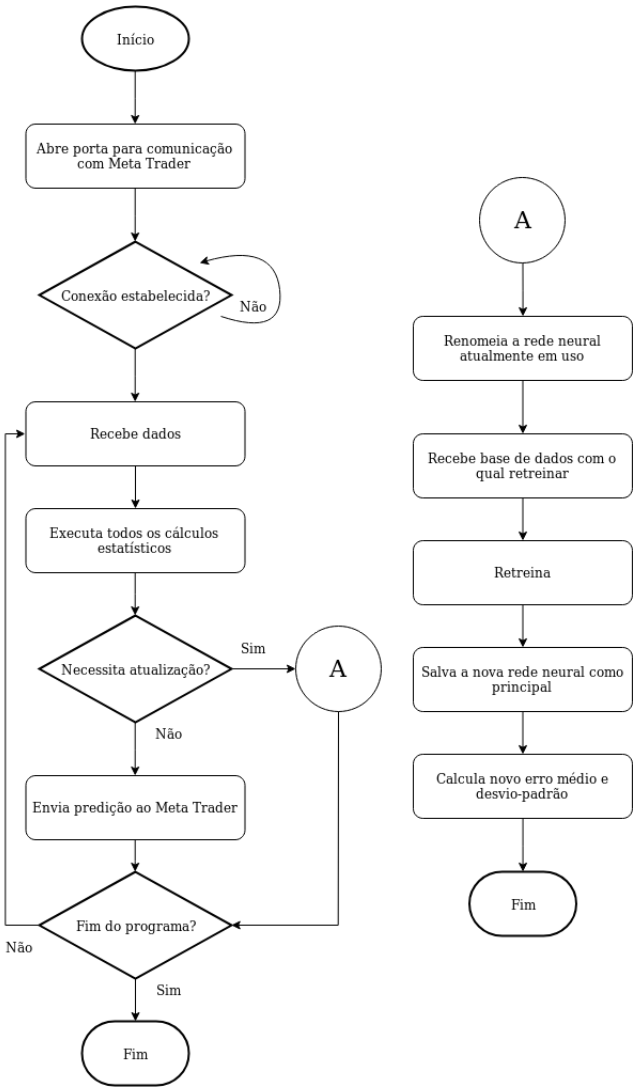
Figura 7 – Diagrama de fluxo da etapa de seleção das redes LSTM com melhor desempenho (Fonte: próprio autor)



3.2.4 Etapa de retreinamento das redes LSTM selecionadas

A descrição em diagrama de fluxo desta etapa é mostrada na Figura 8.a. O objetivo da mesma é facilitar a atualização dos modelos, a fim de reduzir problemas de *concept drift* utilizando um algoritimo capaz de analisar a variação do erro no tempo e caso necessário retreinar a rede LSTM a fim de regressá-la a uma margem aceitável de erro. Para tanto, tal algoritimo faz uso de comunicação *online* com o *Meta Trader* para fazer aquisição de dados em tempo real.

Figura 8 – Diagrama de fluxo da Etapa da etapa de retreinamento das redes LSTM selecionadas (Fonte: próprio autor)



3.3 Metodologia de seleção do modelo de predição

Para distinguir o rendimento das diferentes arquiteturas testadas, decidiu-se por utilizar o MSE por ser sensível à ruído, característica importante, uma vez que séries temporais financeiras são bastante ruidosas. a métrica MSE é definida como:

$$MSE = \frac{\Sigma(predito - real)^2}{Número\ total\ de\ dados} \tag{3.3}$$

Para a classificação da melhor arquitetura levou-se em consideração, portanto, aqueles que apresentaram menor MSE para o conjunto de validação.

Usando a metodologia descrita, nesta etapa foram determinadas três arquiteturas distintas, com horizontes de predição de 1, 3 e 10 constantes de tempo (para o gráfico de 15 minutos), que apresentaram resultados satisfatórios.

Tais arquiteturas estão descritas na Tabela 1. Vale ressaltar que todas as arquiteturas apresentam quantidade de saídas igual ao horizonte de predição e que a camada de entrada de cada célula possui a mesma quantidade de neurônios que a primeira camada oculta, o funcionamento da célula é descrito na seção 2.6.

Na Figura 9 é mostrado um exemplo de organização de uma arquitetura LSTM, com 3 células e horizonte de predição genérico y, com a saída realimentada.

Figura 9 – Representação do modelo LSTM (Fonte: (GRAVES, 2013)

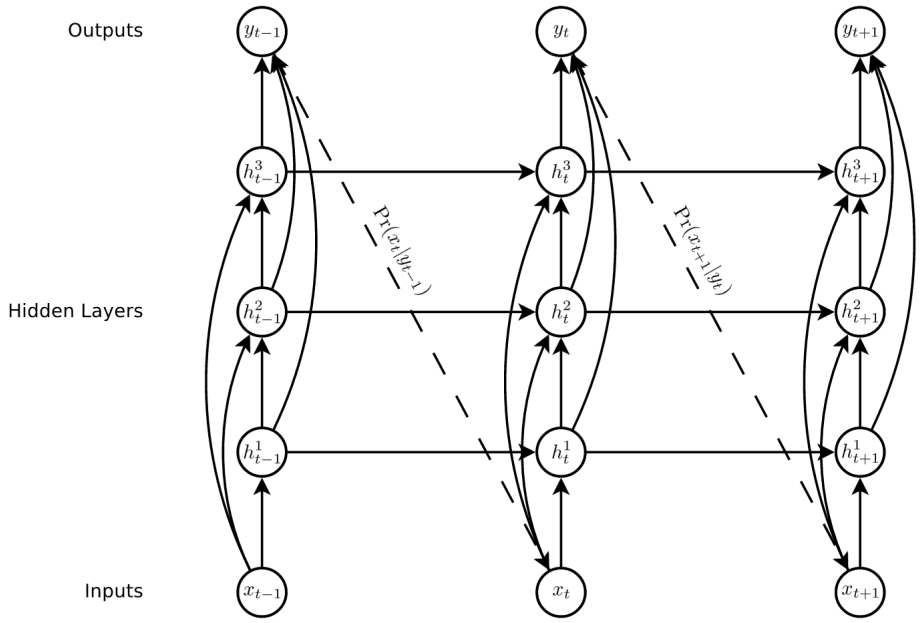


Tabela 1 – Arquitetura das redes neurais utilizadas

Modelo	*Look Back*[4]	Horizonte de predição	Células	Neurônios por camada
Modelo 13	5	1	4	64
Modelo 0	5	3	2	32
Modelo 340	5	10	2	64

[4] Número de instantes de tempos passadas analisados para se fazer a predição

3.4 Pipeline implementado para analise em um ambiente real

Para que fosse possível a implementação do projeto de forma prática, faz-se necessário que o modelo implementado em *Python* tenha acesso a valores de cotação online para que possa fazer as predições e que o *Expert Advisor* do *Meta Trader* tenha acesso ao resultado da predição. Porém o programa *Meta Trader*, por questão de segurança não permite que seus *Advisors* acessem diretórios externos aos seu próprio, além de que o nome do diretório é criado a partir de uma função de Hash[5], e portanto aleatória para cada computador e cliente gerado.

Portanto para superar tais desafios decidiu-se por fazer uma conexão via *socket* entre o *Meta Trader* e o modelo com o intuito de enviar ao modelo o endereço do diretório onde o *Meta Trader* está habilitado a fazer leitura e escrita de arquivos. Uma vez estabelecida a conexão o *Expert Advisor* envia à porta especificada uma *string* correspondendo ao endereço do diretório onde se encontram os arquivos para leitura e escrita, por fim a conexão via *socket* é fechada.

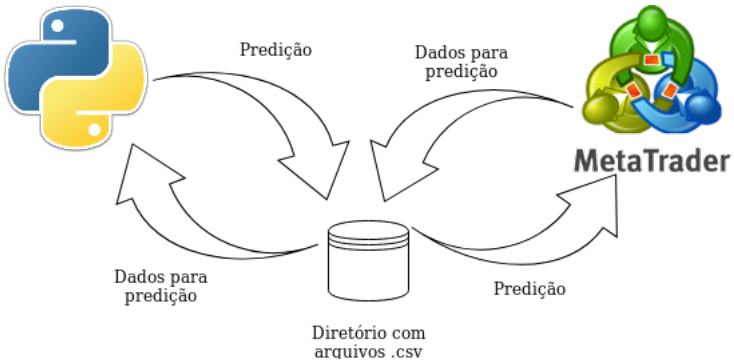
Figura 10 – Exemplificação da conexão *Meta Trader*/Python utilizando *socket* (Fonte: próprio autor)

Uma vez finalizada a conexão via *socket* os programas passam a se comunicar via arquivos, com o auxílio de um algoritimo semáforo[6]. Ainda com o intuito de evitar-se colisões de dados, decidiu-se por utilizar arquivos CSV diferentes para a escrita dos dados passados que serão utilizados pelo preditor e para a escrita da predição em si. Na Figura 11 é possível visualizar um esquemático básico do fluxo de informação.

[5] A função Hash converte uma entrada de letras e números em uma saída criptografada de tamanho fixo (HASH, 2018).

[6] Um semáforo é um objeto de *kernel* que uma ou mais tarefas pode adquirir ou liberar com o propósito de sincronização ou exclusão mútua (RICHARDSON, 2016)

Figura 11 – Exemplificação da comunicação *Meta Trader*/Modelo utilizando arquivos CSV. (Fonte: próprio autor)



3.5 Resumo

Deste capítulo pode-se concluir que apesar da inerente dificuldade em se prever séries temporais financeiras problemas como base de dados confiável e limitação do espaço de busca de hiperparâmetros são obstáculos a serem superados para que seja possível encontrar um modelo que faça uma boa predição. Outro problema de séries temporais financeiras é o fato das mesmas não serem estacionárias e portanto variarem no tempo, o que faz necessário ajustes periódicos nos pesos do modelo. O que segue deste trabalho são análise dos resultados encontrados a partir da utilização do algoritimos aqui explicados.

4 RESULTADOS

4.1 Introdução

Neste capítulo serão apresentados os resultados obtidos, utilizando-se como métricas de análise as características estatísticas do preditor (MSE, erro médio e desvio-padrão do erro), por fim os resultados serão comparados com os métodos ARMA e média móvel simples de 12 períodos. De tal modo o capitulo inicia descrevendo o banco de dados utilizado para treinamento e teste das abordagens propostas, também são apresentados todos os resultados obtidos em diferentes etapas do processo, mostrando a evolução obtida a partir da implementação de algumas técnicas apresentadas anteriormente. E por fim, é feita uma análise destes resultados e uma comparação com os resultados obtidos por outros trabalhos semelhantes com o intuito de validar as abordagens propostas.

4.2 Banco de dados GBP/USD *Robo Forex*

Para a realização do treinamento dos preditores e validação dos resultados é imprescindível uma base de dados consolidada. Portanto utilizou-se a base de dados fornecida pela *Robo Forex*.

A base de dados abrange todos os dados em intervalo de 15 minutos do 1 de julho de 2013 até 18 de julho de 2018, portanto 5 anos de informação disponível para uso. Devido à não estacionariedade das séries temporais, decidiu-se por não utilizar toda a base de dados,visto que dados tão antigos não teriam influência nos resultados. Para o modelo 340 utilizou 80.000 pontos enquanto que para os modelos 0 e 13 utilizou-se 40.000 pontos. A base de dados apresenta como parâmetros de entrada os valores de abertura, máxima, mínima e fechamento de cada intervalo de tempo.

Os dados utilizados para treinar as redes foram divididas em três conjuntos: treinamento, validação e teste. O conjunto de treino possui 80% dos dados, validação e treino 10% cada.

A imputação de dados faltantes se deu pelo método de *feed foward*, apesar de que a base de dados apresentava $99,90\%$ dos dados completos e portanto a quantidade de dados imputados é mínima, sendo assim, não tem influência considerável no resultado final.

4.3 Recursos Computacionais

Recursos de *Software*. A aquisição dos dados de mercado foi feita através do programa *Meta Trader* conectado ao servidor de uma corretora de câmbio, no caso Robo *Forex*, associada a uma conta demonstrativa. A etapa de limpeza e organização dos dados, assim como o cálculo e análise de dados estatísticos e o desenvolvimento de gráficos foi feita em Python. As etapas de treinamento e teste da arquitetura proposta foram realizadas utilizado o API *Keras*, *software* de código aberto desenvolvido por toda uma comunidade ativa [1], criado com o foco em possibilitar desenvolvimentos rápidos, buscando fácil usabilidade, extensibilidade e modularidade.

Após as etapas de treinamento e teste da arquitetura foi desenvolvido um algoritimo em *MQL5* de forma a possibilitar a utilização do *Expert Advisor* em uma plataforma de negociação real.

Recursos de *Hardware*. O computador utilizado nos experimentos possuía a seguinte configuração: (*i*) sistema operacional Linux, distribuição Ubuntu Server 16.04; (*ii*) processador Intel Core i7-7700, 3.60GHz com 4 núcleos físicos; (*iii*) memória RAM de 16 GB; (*iv*) unidade de armazenamento de 1TB (disco rígido); (*v*) placa de vídeo Nvidia Geforce GTX 1080, com 8 GB de memória dedicada. Além disso, um computador pessoal foi utilizado em alguns experimentos, tal computador possuía a seguinte configuração: (*i*) sistema operacional Linux, distribuição Ubuntu 18.04.1 LTS; (*ii*) processador Intel Celeron B830, 1.80GHz com 2 núcleos físicos; (*iii*) memória RAM de 8 GB; (*iv*) unidade de armazenamento de 500GB (disco rígido);

4.4 Detalhes de implementação

Como a capacidade da memória RAM não era suficiente para realizar todos os treinos de todos os modelos de uma só vez, fez-se necessário treinar somente 100 arquiteturas por vez. Ao final de 500 arquiteturas testadas, concluiu-se que já não se fazia necessário outros testes, pois os resultados encontrados eram satisfatórios.

Os hiperparâmetros que apresentaram os melhores resultados para as redes neurais descritas são apresentados na Tabela 2. Os demais hiperparâmetros foram apresentados na Tabela 1. A busca por esses hiperparâmetros foi empírica. O fato de todas a redes possuírem os

[1] <https://keras.io/>

vários hiperparâmetros idênticos é resultado de coincidência, uma vez que todas as redes foram submetidas a uma série de combinações dos mesmos.

Tabela 2 – Hiper parâmetros das redes neurais utilizadas

Dropout	0.5
Tamanho de *Batch*	1024
Janela de análise	5
Otimizador	Adam

4.5 Experimentos

4.5.1 Métricas

Uma vez selecionadas as redes a se utilizar, os resultados serão analisados segundo as seguintes métricas: (i) correlação de Pearson; (ii) média e desvio padrão do erro. Abaixo são explicados brevemente tais métricas:

- **Correlação de Pearson**. Correlação é uma técnica de investigação de relacionamento entre duas unidades quantitativas. O coeficiente de correlação de Pearson (r), é uma medida de força de associação entre duas variáveis. Uma correlação positiva indica que ambas as variáveis incrementam ou decrementam juntas, enquanto que uma correlação negativa indica que enquanto uma incrementa a outra decrementa, e vice-versa (ENGLAND, 2018). Os valores de correlação de Pearson variam de -1 a $+1$, sendo que -1 representa correlação perfeitamente negativa e $+1$ correlação perfeitamente positiva.
- **Desvio-padrão**. O desvio padrão determina a dispersão dos valores em relação à média e é calculado por meio da raiz quadrada da variância. Onde a variância é a média aritmética dos quadrados dos desvios em torno da média (CORREA, 2003).

4.5.2 Avaliação dos modelos selecionados

Nesta seção será feita a análise dos resultados das arquiteturas desenvolvidas, avaliando suas performances utilizando as métricas discutidas anteriormente.

- **Modelo 0**. Este modelo faz uso de uma janela de 5 constantes de tempo para um horizonte de predição de três intervalos de tempo. Para o mesmo utilizou-se uma

base de dados com 40.000 dados temporais em intervalos e 15 minutos, equivalente a aproximadamente 1,5 anos de dados, dos quais 80% foi utilizado para treino, 10% para validação e os restantes 10% para teste.

Na Figura 12 são apresentados os gráficos de correlação e erro da rede para o conjunto de teste. Tal gráfico mostra que a predição tem uma correlação quase perfeita com os dados reais, apresentando uma média e desvio padrão de erro pequenos. Especificamente o erro de predição apresenta um erro médio de -0.00016 com desvio padrão de 0.00119.

Na Figura 13 se observa a eficácia do modelo 0 em predizer os dados para o conjunto de teste, que compreende de 4000 pontos. É possível observar que os maiores picos de erro ocorrem em situação onde há uma variação brusca no valor do ativo.

Figura 12 – Modelo 0: Análise dos resultados estatísticos

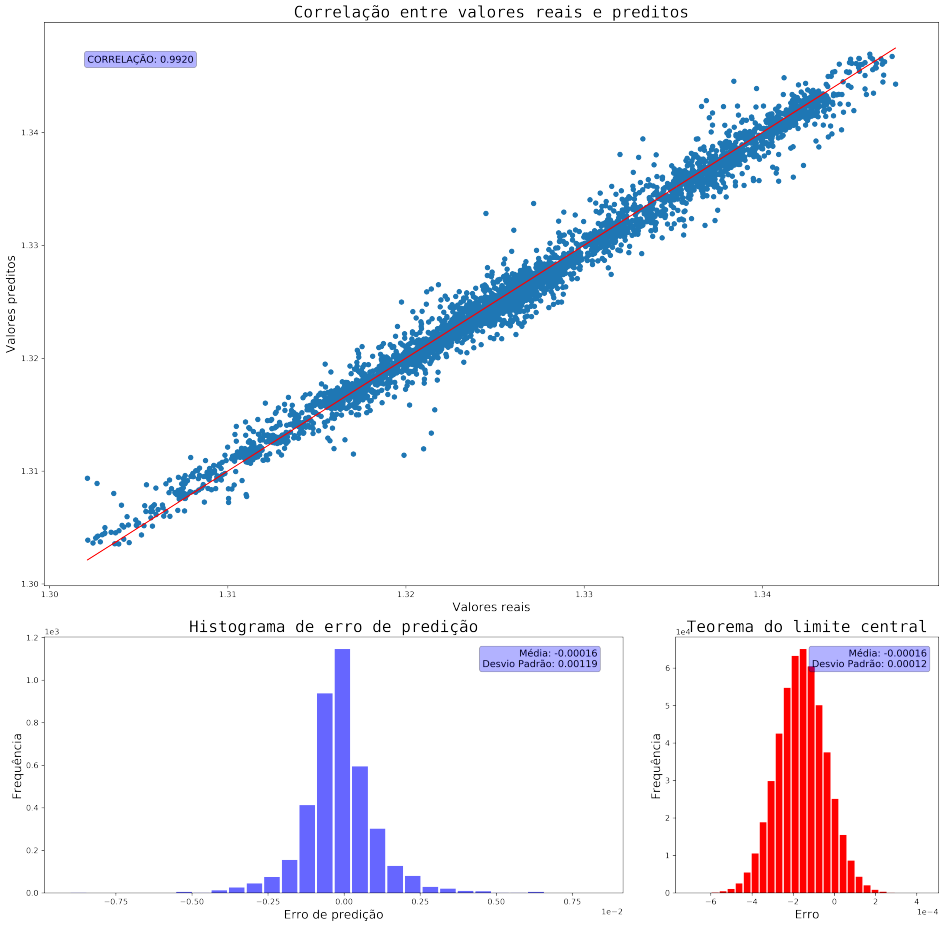


Figura 13 – Modelo 0: Comparação real × predito

- **Modelo 13**. Este modelo faz uso de uma janela de 5 constantes de tempo (dados passados) para um horizonte de predição de um horizonte de tempo. Para o mesmo utilizou-se uma base de dados com 40.000 dados temporais em intervalos e 15 minutos, equivalente a aproximadamente 1,5 anos de dados, dos quais 80% foi utilizado para treino, 10% para validação e os restantes 10% para teste.

  Na Figura 14 são apresentados os gráficos de correlação e erro da rede para o conjunto de teste. Tal gráfico mostra que a predição tem uma correlação forte correlação com os dados reais, apresentando uma média e desvio padrão de erro pequenos. Especificamente o erro de predição apresenta um erro médio de 0.00042 com desvio padrão de 0.00111.

  Na Figura 15 se observa a eficácia da modelo 13 em predizer os dados para o conjunto de teste, que compreende de 4.000 pontos. É possível observar que os maiores picos de erro ocorrem em situação onde há uma variação brusca no valor do ativo.

Figura 14 – Modelo 13: Análise dos resultados estatísticos

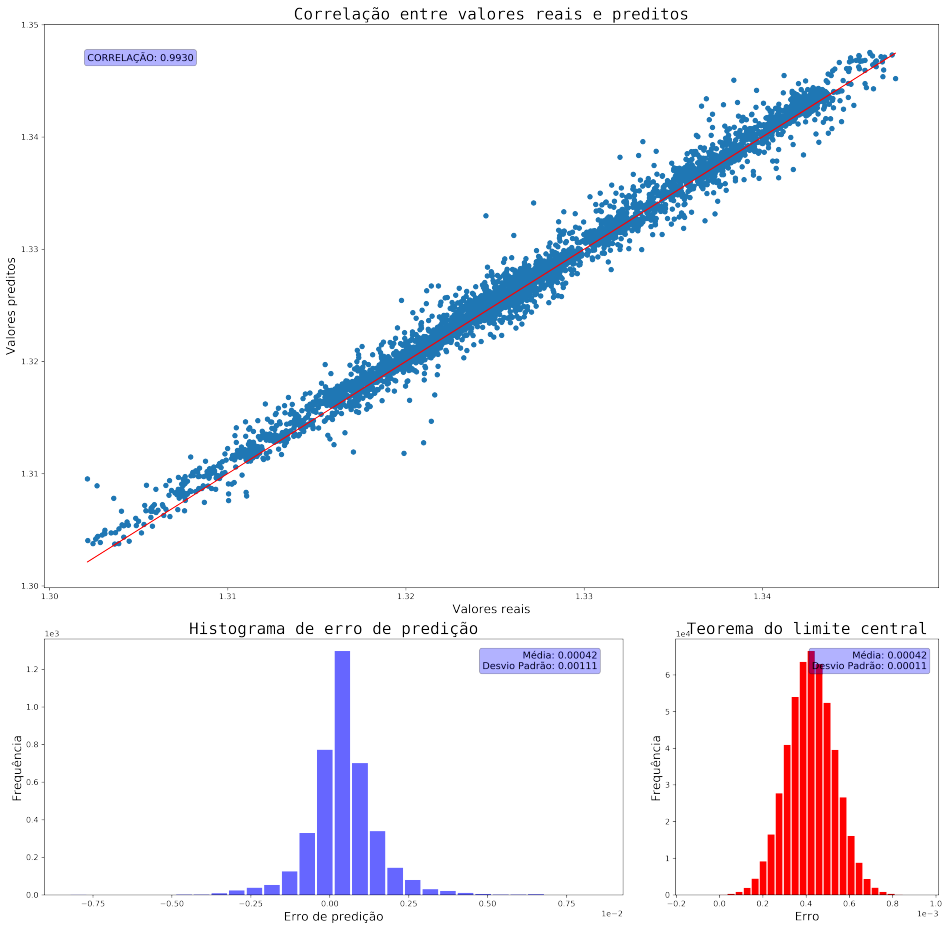


Figura 15 – Modelo 13: Comparação real × predito

- **Modelo 340**. Este modelo faz uso de uma janela de 5 constantes de tempo para um horizonte de predição de dez intervalos de tempo. Para o mesmo utilizou-se uma base de dados com 80.000 dados temporais em intervalos e 15 minutos, equivalente a aproximadamente 3 anos de dados, dos quais 80% foi utilizado para treino, 10% para validação e os restantes 10% para teste.

  Na Figura 16 são apresentados os gráficos de correlação e erro da rede para o conjunto de teste. Tal gráfico mostra que a predição tem uma correlação quase perfeita com os dados reais, porém apresentando uma média e desvio padrão altos quando comparados com os demais modelos, isso se deve também ao aumento de dificuldade em prever tão longinquamente. Especificamente o erro de predição apresenta um erro médio de -0.00173 com desvio padrão de 0.00269.

  Na Figura 17 se observa a eficácia do modelo 340 em predizer os dados para o conjunto de teste, que compreende de 8.000 pontos. É visível que o modelo, nos primeiros dois terços das amostras apresenta um valor de predição abaixo do valor real, porém a partir do terceiro terço o mesmo passa a predizer com menores erros.

Figura 16 – Modelo 340: Análise dos resultados estatísticos

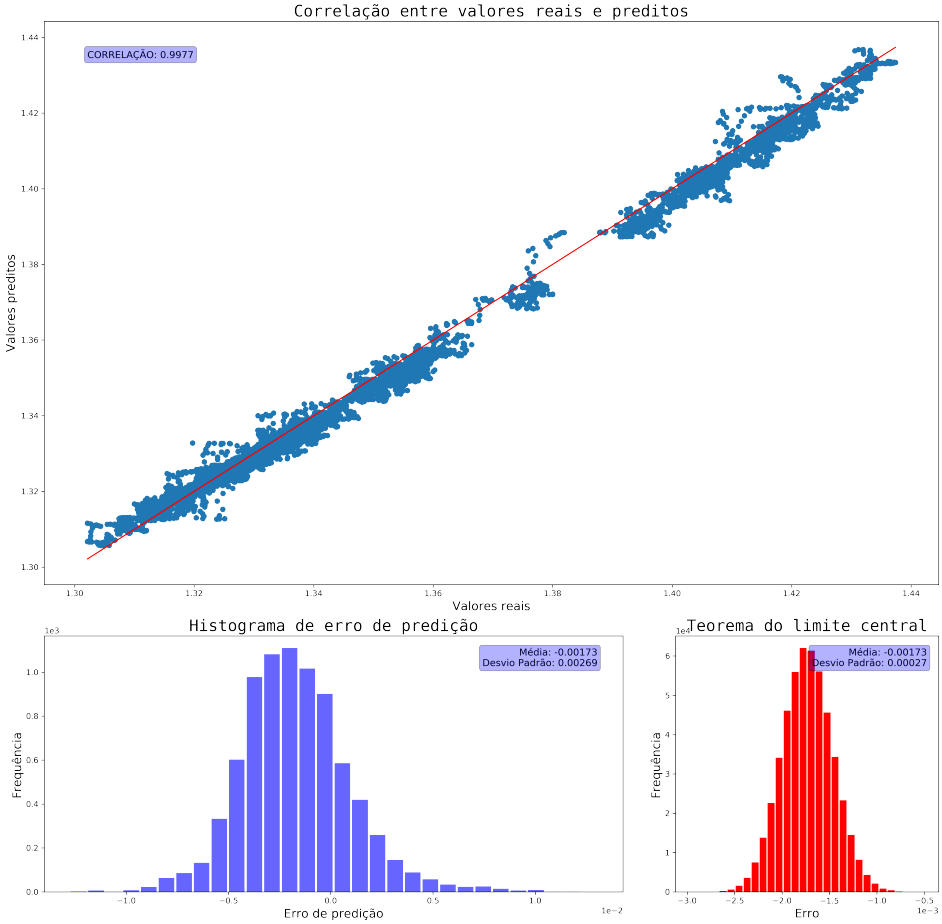


Figura 17 – Modelo 340: Comparação real × predito

Na Tabela 3 abaixo é possível analisar os resultados segundo as métricas descritas na seção 4.5.1, utilizando-se o conjunto de testes.

Tabela 3 – Resultados estatísticos

Modelo	Correlação	Média	Desvio padrão
Modelo 0	0.9920	-0.00016	0.00119
Modelo 13	0.9930	0.00042	0.00111
Modelo 340	0.9971	-0.00173	0.00269

Como base de comparação foi utilizado modelo de média móvel e ARMA para cada um dos horizontes de predição. Para o modelo ARMA utilizou-se um modelo de primeira ordem, uma vez que modelos de ordem maiores requerem que a matriz de entrada seja inversível, o que não é o caso para a base de dados em questão. Para o modelo de média móvel utilizou-se um intervalo de 5 instantes de tempo, equiparando-se portanto à quantidade de células das redes neurais LSTM.

Tabela 4 – Comparação de resultados com modelos lineares

	Modelo 13			
	MSE	Erro Médio	Desvio padrão	Correlação
Neural LSTM	3.87E-06	0.00042	0.00111	0.993
ARMA	6.95E-05	0.00182	0.00190	0.721
Média Móvel	1.66E-05	0.01549	0.00681	0.982
	Modelo 0			
	MSE	Erro Médio	Desvio padrão	Correlação
Neural LSTM	3,97E-06	0,00016	0,00119	0,992
ARMA	1,02E-04	0,00971	0,00269	0,894
Média Móvel	3,51E-05	0,01572	0,00682	0,962
	Modelo 340			
	MSE	Erro Médio	Desvio padrão	Correlação
Neural LSTM	1,97E-05	0,00173	0,00269	0,997
ARMA	5,25E-04	0,01645	0,00706	0,475
Média Móvel	6,78E-05	0,01518	0,00682	0,926

4.5.3 Resultados de *Backtest*

A fim de validar o uso dos modelos no mercado de ForEx, faz-se necessário realizar um *backtest*, que consiste em analisar o retorno da estratégia a partir de dados passados. Portanto foram realizados, para cada modelo, 2 *backtests*, ambos com *stop loss*[2] de 100 pontos[3], porém com *take profit*[4] de 100 e 300 pontos. A abertura de operação se dá caso o valor predito tenha uma diferença mínima de 200 pontos com o valor atual. Como base de comparação utilizou-se um método de rompimento, cujos limiares de compra e venda encontravam-se 100 pontos acima e abaixo do valor presente. O intervalo de análise foi de 19/11/2018 a 26/11/2018. O resultado pode ser analisado na Tabela 5.

Tabela 5 – Resultados de *backtest*

	Take profit 100				Take profit 300			
	Lucro	*Drawdown*	Acurácia	$E[x]$	Lucro	*Drawdown*	Acurácia	$E[x]$
Rompimento	-842	1089	38.24%	-23.47	-1011	1962	18.75%	-25
Modelo 13	1800	500	65.62%	31.24	3500	800	40.32%	61.28
Modelo 0	1150	600	54.63%	9.26	3100	1000	37.50%	50
Modelo 340	-1800	2000	44.10%	-11.8	-800	2000	25.53%	2.12

[2] Perda máxima aceitável para a operação deva ser finalizada
[3] Menor variação do ativo em questão
[4] Ganho com o qual a operação deva ser encerrada

Figura 18 – Backtest Modelo 0 - Take profit 100

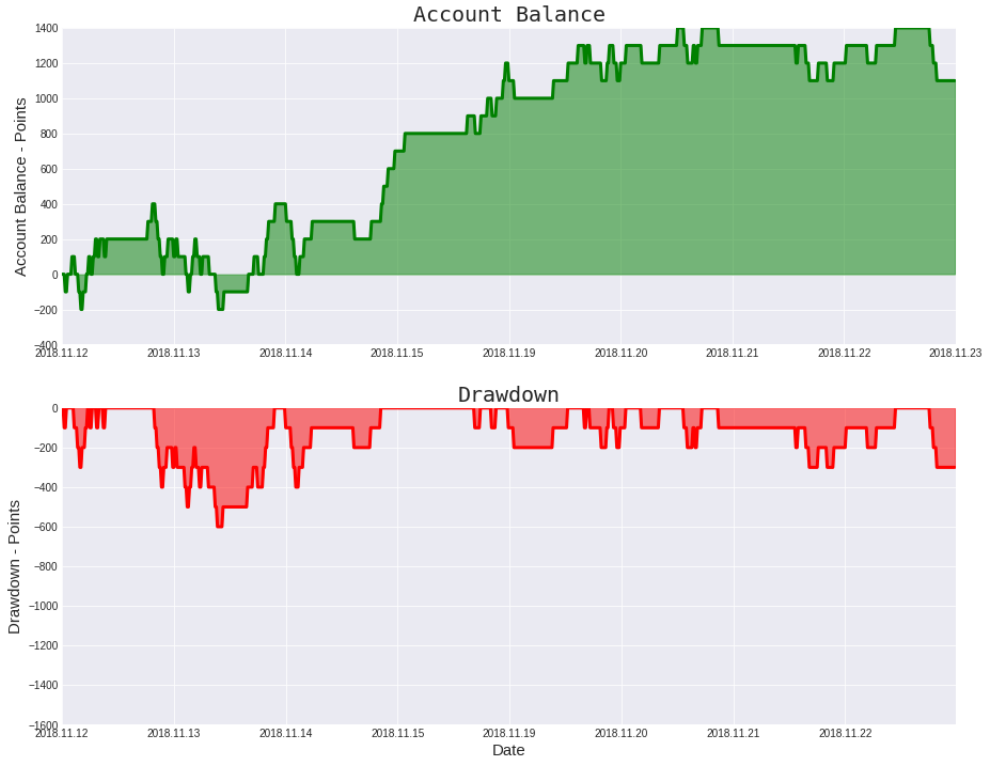


Figura 19 – Backtest Modelo 0 - Take profit 300

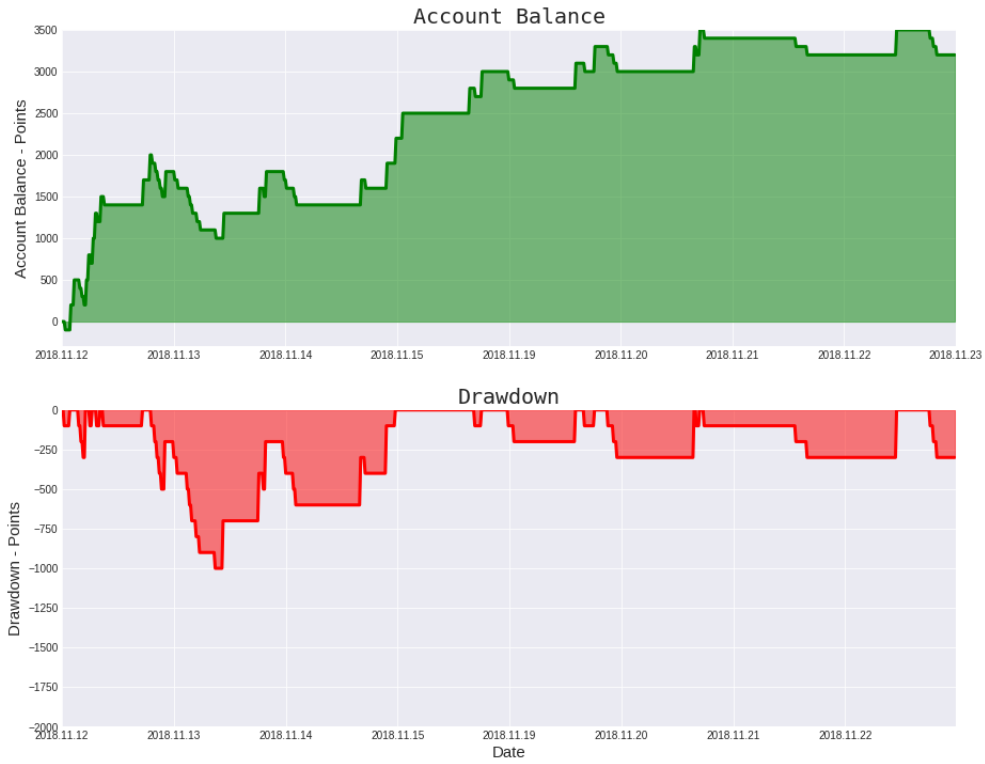


Figura 20 – Backtest Modelo 13 - Take profit 100

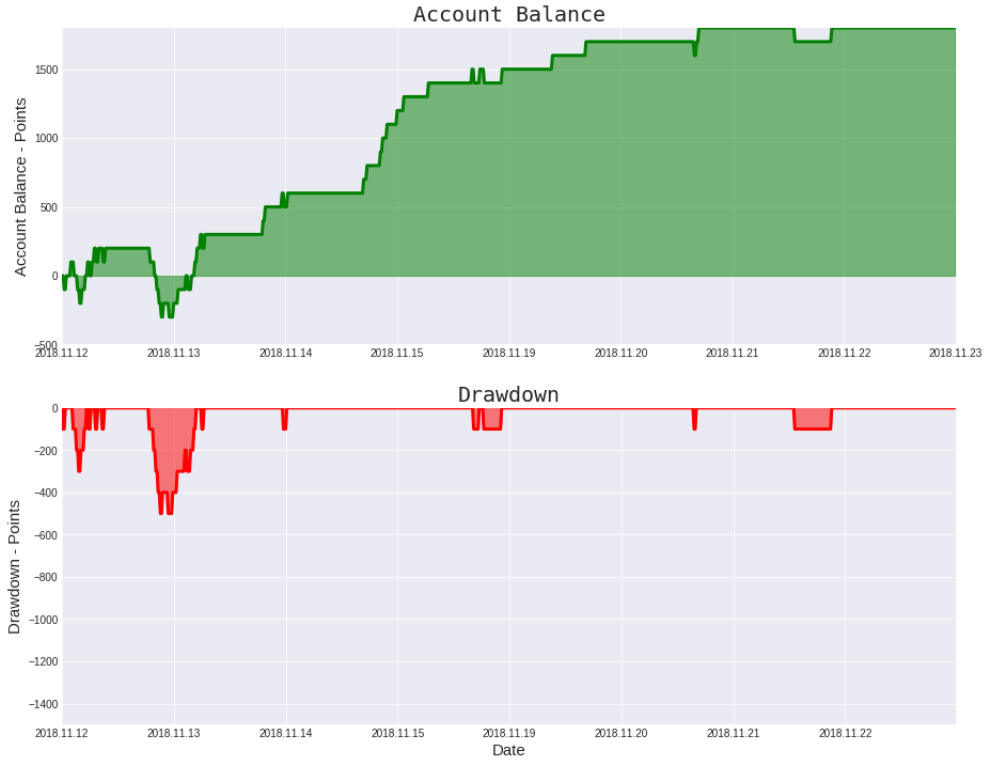


Figura 21 – Backtest Modelo 13 - Take profit 300

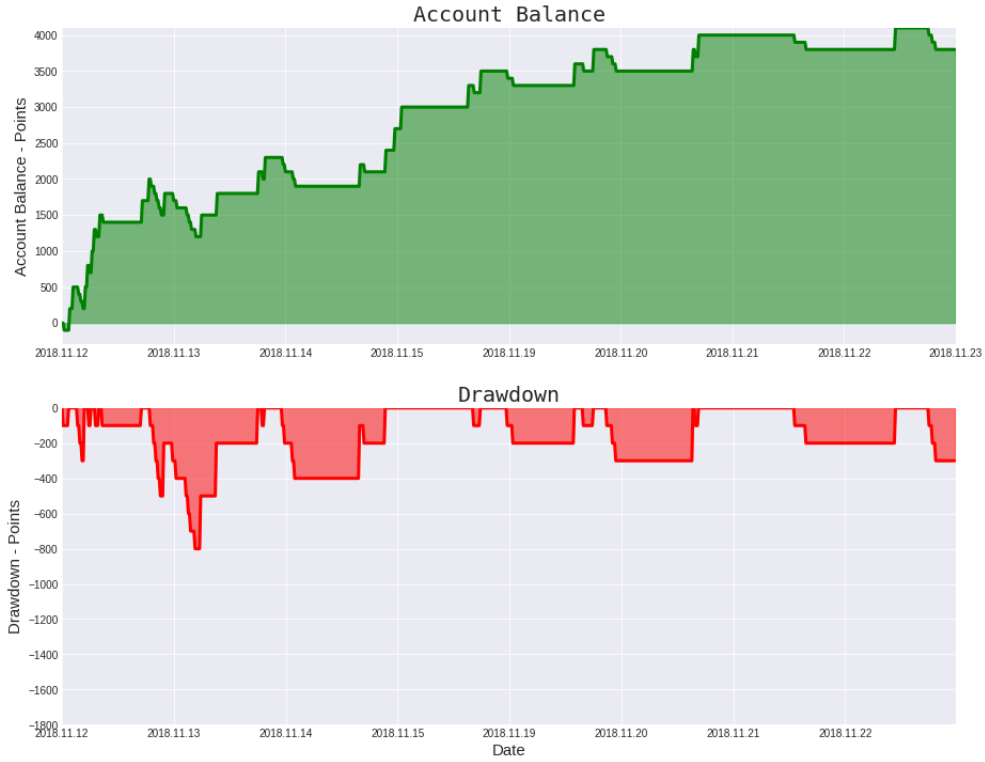


Figura 22 – Backtest Modelo 340 - Take profit 100

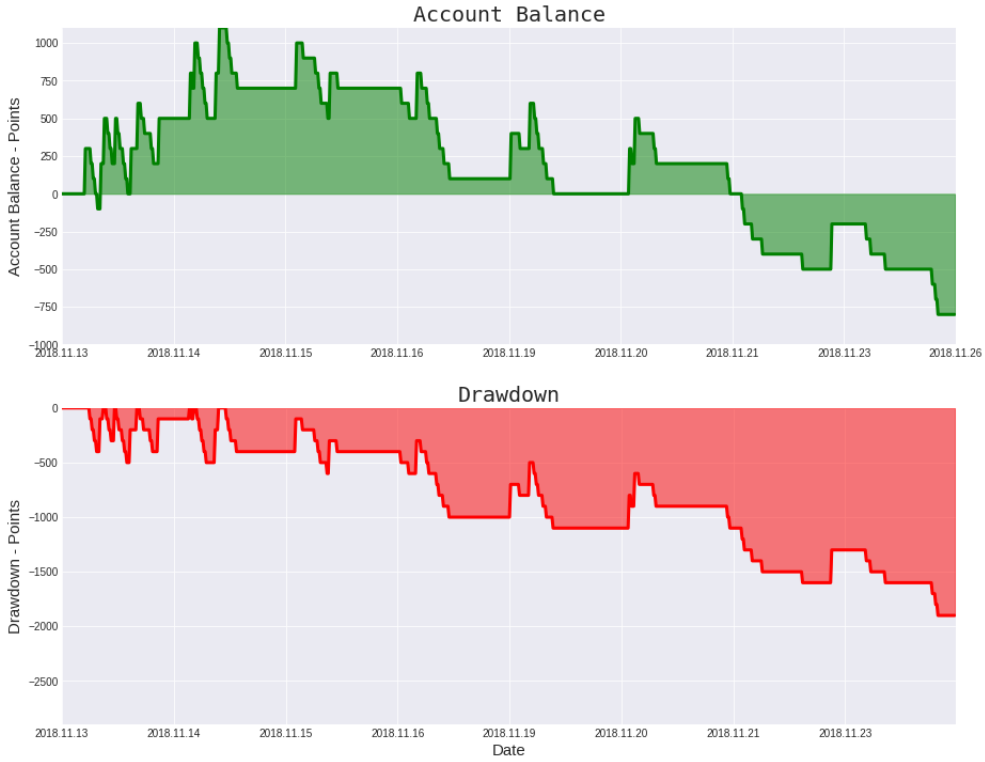
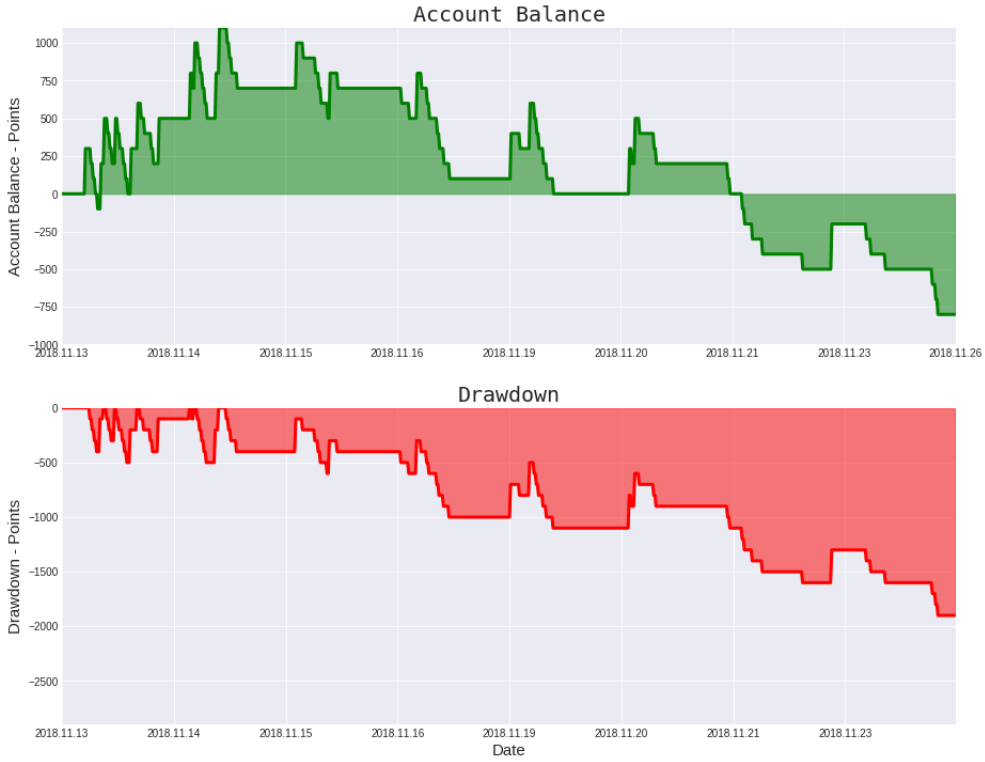


Figura 23 – Backtest Modelo 340 - Take profit 300



4.6 Resumo

Neste capítulo foram analisados os resultados dos modelos desenvolvidos para o par de moedas GBPUSD e por fim feitas as comparações com modelos lineares mais comumente utilizados. Foi possível, portanto, observar a superioridade da rede neural em comparação com modelos ARMA e de médias móveis, resultando em uma precisão até 30 vezes melhor que tais modelos. Portanto pode-se afirmar que modelos não lineares, como redes neurais têm maior capacidade de abstração e previsão para séries estocásticas, como é o caso de séries temporais financeiras.

5 CONCLUSÕES E PROJETOS FUTUROS

5.1 Conclusões

O objetivo principal do trabalho foi propor um modelo de capaz de prever variações do ativo GBPUSD no mercado de ForEx. A técnica proposta é baseada fundamentalmente em redes neurais recorrentes, especificamente redes LSTM. Tal abordagem é motivada pela vantagem de sua adaptação no tempo e a não existência dos problemas de desaparecimento e explosão de gradiente, ambos problemas inerentes de arquiteturas de redes neurais profundas.

Como resultado, usando um *look back* de 5 intervalos de tempos foi possível encontrar redes neurais LSTM capazes de fazer previsões validas para 1, 3 e 10 horizontes de tempo. Com erros médio e desvio padrão pequenos em relação aos modelos com os quais foram comparados.

A partir dos valores das métricas encontradas é possível afirmar que redes neurais LSTM apresenta uma valida capacidade de previsão, e resulta em modelos fortemente correlacionados com os valores reais, além de apresentar erros médio e desvios padrão pequenos quando comparados com métodos tradicionais.

Portanto a utilização de redes neurais demonstra-se como boa alternativa na utilização como indicador de tendência para ativos financeiros de alta liquidez, como é o caso do GBPUSD.

5.2 Temas a serem pesquisados

Como trabalho futuro podemos citar:

Análise detalhada dos resultados quanto à sua operação *online*, assim como a validação do algoritimo de retreino como solução para o problema de *concept drift*.

Inserir também como entrada valores de volatilidade, volume e outros indicadores técnicos tradicionalmente utilizados no mercado financeiro, a fim de prover mais informação à rede neural.

Em complemento à rede neural faz-se necessário o desenvolvimento de uma estratégia de *trading* que faça bom uso das informações providas pelo modelo, a mesma deve ser testada junto a um conjunto de teste para análise de retorno, fator de lucro e *drawdown*[1].

Portanto uma análise metódica dos sistemas devem ser feito, através de contas demonstrativas antes de se pensar em investir com contas reais. Uma busca mais profunda na parametrização pode revelar modelos mais atrativos e com melhor performance, assim como a associação de vários modelos.

[1] Maior distância entre um topo e um fundo subsequente no gráfico de balanço

REFERÊNCIAS

BRASIL, B. C. do. FAQ - Câmbio - Mercado de câmbio - definições. [S.l.]: Banco Central do Brasil, 2018. Https://www.bcb.gov.br/pre/bc$_atende/port/merccam.asp.Accessed$: $2018 - 09 - 28.Citadonapágina12$.

BRASIL, B. C. do. Produto Interno Bruto e taxas médias de crescimento. 2018. <https://www.bcb.gov.br/pec/Indeco/Port/indeco.asp>. Accessed: 2018-09-26. Citado na página 12.

BROWNLEE, J. A Gentle Introduction to Concept Drift in Machine Learning. 2018. <https://machinelearningmastery.com/gentle-introduction-concept-drift-machine-learning/>. Accessed: 2018-11-23. Citado na página 24.

CORREA, S. M. B. B. Probabilidade e estatística. [S.l.]: PUC Minas Virtual, 2003. 61 p. Citado na página 34.

DEFINITION of spread. [S.l.]: Financial Times, 2018. Http://lexicon.ft.com/Term?term=spread. Accessed: 2018-09-27. Citado na página 16.

ENGLAND, U. of the West of. Pearson's Correlation Coefficient. 2018. <http://learntech.uwe.ac.uk/da/Default.aspx?pageid=1442>. Citado na página 34.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. Deep Learning. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>. Citado 3 vezes nas páginas 7, 17 e 19.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. Deep Learning. [S.l.]: MIT Press, 2016. Citado na página 13.

GRAVES, A. Generating sequences with recurrent neural networks. CoRR, abs/1308.0850, 2013. Disponível em: <http://dblp.uni-trier.de/db/journals/corr/corr1308.html#Graves13>. Citado 2 vezes nas páginas 7 e 29.

GREFF, K. et al. Lstm: A search space odyssey. IEEE Transactions on Neural Networks and Learning Systems, v. 28, n. 10, p. 2222–2232, Oct 2017. ISSN 2162-237X. Citado na página 13.

HANSSON, M. On stock return prediction with LSTM networks. 2017. Student Paper. Citado na página 14.

HASH. [S.l.]: Investopedia, 2018. Https://www.investopedia.com/terms/h/hash.asp. Accessed: 2018-09-27. Citado na página 30.

HE, K. et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision. [S.l.: s.n.], 2015. ISBN 9781467383912. ISSN 15505499. Citado na página 13.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. Neural Computation, v. 9, n. 8, p. 1735–1780, 1997. Disponível em: <https://doi.org/10.1162/neco.1997.9.8.1735>. Citado 2 vezes nas páginas 13 e 20.

INVESTOPEDIA. Buy and Hold. [S.l.]: Investopedia, 2018. Https://www.investopedia.com/terms/b/buyandhold.asp. Accessed: 2018-09-27. Citado na página 14.

Lipton, Z. C.; Berkowitz, J.; Elkan, C. A Critical Review of Recurrent Neural Networks for Sequence Learning. ArXiv e-prints, maio 2015. Citado na página 20.

MAINDONALD, J. H. Time series analysis with applications in r, second edition by jonathan d. cryer, kung-sik chan. International Statistical Review, v. 77, n. 2, p. 300–301, 2009. Disponível em: <https://EconPapers.repec.org/RePEc:bla:istatr:v:77:y:2009:i:2:p:300-301>. Citado na página 16.

NASCIMENTO, R. C. Avaliação de oportunidades de investimento no mercado futuro brasileiro na escala de dezenas de segundos. 03 2018. Citado na página 14.

NASDAQ. Efficient Market Hypothesis. 2018. <https://www.nasdaq.com/investing/glossary/e/efficient-market-hypothesis>. Accessed: 2018-11-22. Citado na página 14.

NELSON, D. M. Q.; PEREIRA, A. C. M.; OLIVEIRA, R. A. de. Stock market's price movement prediction with lstm neural networks. In: 2017 International Joint Conference on Neural Networks (IJCNN). [S.l.: s.n.], 2017. p. 1419–1426. ISSN 2161-4407. Citado na página 14.

OLAH, C. Understanding LSTM Networks. 2018. <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>. Citado 3 vezes nas páginas 7, 21 e 22.

OPENAI. Dota 2. 2017. Citado na página 13.

PITTS, W.; MCCULLOCH, W. S. How we know universals the perception of auditory and visual forms. The bulletin of mathematical biophysics, v. 9, n. 3, p. 127–147, Sep 1947. ISSN 1522-9602. Disponível em: <https://doi.org/10.1007/BF02478291>. Citado 3 vezes nas páginas 7, 17 e 18.

RICHARDSON, T. Real time programming - lecture 08: Semaphores. 09 2016. Citado na página 30.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, p. 65–386, 1958. Citado na página 17.

SCHMIDHUBER, J. Deep learning in neural networks: An overview. Neural Networks, v. 61, p. 85 – 117, 2015. ISSN 0893-6080. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0893608014002135>. Citado na página 19.

SEJNOWSKI, T. The prophet who foretold our future. Brain, p. awy212, 2018. Disponível em: <http://dx.doi.org/10.1093/brain/awy212>. Citado na página 17.

SETTLEMENTS, B. of I. Triennial Central Bank Survey - Foreign exchange turnover in April 2016. 2016. Citado 2 vezes nas páginas 12 e 23.

SILVER, D. et al. Mastering the game of Go with deep neural networks and tree search. Nature, Nature Publishing Group, a division of Macmillan Publishers Limited. All Rights Reserved., v. 529, p. 484, jan 2016. Citado na página 13.

STOPPIGLIA, H.; IDAN, Y.; DREYFUS, G. Neural-network-aided portfolio management. 06 1998. Citado na página 13.

Anexos

.1 código Main

```
"""
+------------------------------------------------------------------------------+
|                                                                MAIN CODE 1.0 |
|                                                     GABRIEL MOUZELLA SILVA |
+------------------------------------------------------------------------------+
@author: Gabriel Mouzella Silva
@copyright: Gabriel Mouzella Silva
"""

#!/usr/bin/env python
# -*- coding: utf-8 -*-

import gc
import logging as lginbr
import data_mining1 as dtm
import model as md
import resume as rs
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from results import after_tests
from mail import mail_sender
import resource
from keras.callbacks import TerminateOnNaN, ModelCheckpoint



def seno():
    fs = 20000 # sample rate
    f = 20 # the frequency of the signal

    x = np.arange(fs) # the points on the x axis for plotting
    # compute the value (amplitude) of the sin wave at the for each sample
    y = [ np.sin(2*np.pi*f * (i/fs)) for i in x]
    return pd.DataFrame({'GBPUSD_OPEN': y, 'GBPUSD_HIGH': y, 'GBPUSD_LOW': y,
                         'GBPUSD_CLOSE': y,})



NOME_DIR = input('Nome do diretrio onde salvar: ')
```

```
#+-----------------------------------------------------------------------------+
#-- CLASS HYPERPARAMETERS                                                      |
#+-----------------------------------------------------------------------------+

class Hyperparameters:
  def __init__(self):
      self.dropout = 0.25
      self.batch_size = 128
      self.epochs = 50
      self.output_size = 5
      self.neurons = [64, 64, 64, 64]
      self.window_len = 15
      self.training_size = 0.8
      self.test_size = 0.2
      self.activ_func = 'tanh'
      self.loss = 'mse'
      self.optimizer = 'adam'
      self.name = 'EURUSD'
      self.layers = 4
      self.file_path = ''
      self.score = 0
      self.metrics = ['mse','mae']
      self.minvalue = 0
      self.maxvalue = 0
      self.punnishment = 0.001

#+-----------------------------------------------------------------------------+
#-- HYPERPARAMETERS VECTORIZATION                                              |
#+-----------------------------------------------------------------------------+
EPOCHS = [400]
NEURONS = [[32,32],[64,64],[96,96]]
BATCH_SIZE = [1024,512,256]
DROPOUT = [0.25,0.5,0.75]
OUTPUT_SIZE = [1,3,10]
WINDOW_LEN = [5,10,15]
NEURONS = [2**i for i in range(9,12+1)]

#+-----------------------------------------------------------------------------+
#-- HYPERPARAMETERS INITIALIZATION                                             |
#+-----------------------------------------------------------------------------+
# getting data
```

```
df,name, MinValues, MaxValues = dtm.datamining()

df = df.drop(['DATE'],1)

df.dropna(axis = 0, inplace = True)
score = []

num_ite = len(EPOCHS)*len(NEURONS)*len(BATCH_SIZE)*len(DROPOUT)\
         *len(OUTPUT_SIZE)*len(WINDOW_LEN)
print('------------------------------')
print('NMERO DE  INTERAES : ' + str(num_ite))
#+-----------------------------------------------------------------------------+
#-- DATA AQUISITION AND CLEANING                                              |
#+-----------------------------------------------------------------------------+
print('------------------------------')
msg = 'TREINAMENTO NMERO {:s}/{:s}:' \
      '\n------------------------------'\
      '\n\tEPOCHS:      {:s}  '\
      '\n\tDROUPOUT:    {:s} '\
      '\n\tBATCH SIZE:  {:s} '\
      '\n\tOUTPUT SIZE: {:s} '\
      '\n\tWINDOW LENGHT: {:s} '\
      '\n\tNEURONS:     {:s}'\
      '\n------------------------------'
COUNTER = 0

if __name__ == "__main__":
    params = Hyperparameters()
    params.training_size = 0.8
    params.test_size = 0.1
    params.name = name
    params.punnishment = 0.0
    params.loss = 'mse'


for output_size in OUTPUT_SIZE:
    params.output_size = output_size

    for window_len in WINDOW_LEN:
        params.window_len = window_len

        df = seno()
```

```
MinValues['GBPUSD_OPEN'], MinValues['GBPUSD_HIGH'] = -1,-1
MinValues['GBPUSD_LOW'], MinValues['GBPUSD_CLOSE'] = -1,-1
MaxValues['GBPUSD_OPEN'], MaxValues['GBPUSD_HIGH'] = 1,1
MaxValues['GBPUSD_LOW'], MaxValues['GBPUSD_CLOSE'] = 1,1

X_train,Y_train,\
X_test,Y_test, X_validation, \
Y_validation = dtm.LSTM_data(df,params, MinValues,MaxValues)

params.minvalue = MinValues[params.name + '_CLOSE']
params.maxvalue = MaxValues[params.name + '_CLOSE']

gc.collect()
# converting to numpy array
X_train, X_test = dtm.to_array(X_train), dtm.to_array(X_test)
Y_train, Y_test = dtm.to_array(Y_train), dtm.to_array(Y_test)
for dropout in DROPOUT:
    params.dropout = dropout

    for batch_size in BATCH_SIZE:
        params.batch_size = batch_size

        for epochs in EPOCHS:
            params.epochs = epochs

            for neurons in NEURONS:
                gc.collect()
                params.neurons = neurons
                params.layers = len(neurons)
                print(msg.format(str(COUNTER+1),str(num_ite),
                                 str(epochs),str(dropout),
                                 str(batch_size),str(output_size),
                                 str(window_len),str(neurons)))
                COUNTER += 1

                params.file_path, directory = rs.mkDir(NOME_DIR)

                best_model = params.file_path + 'best_model.h5'
                callbacks = [ModelCheckpoint(best_model,
                                             monitor = 'val_mse',
                                             save_best_only = True,
                                             mode = 'min', period = 1),
```

```
            TerminateOnNaN()]

# initialise model architecture
model = md.build_model(X_train, params)
# train model on data

history = model.fit(X_train, Y_train,
                    epochs=params.epochs,
                    batch_size=params.batch_size,
                    verbose=1,
                    validation_data=(X_test, Y_test),
                    shuffle=False,
                    callbacks = callbacks)

if (params.test_size != 0):
    params.score =
        model.evaluate(X_validation,Y_validation)
    y_real = Y_validation
else:
    params.score = model.evaluate(X_test,Y_test)
    X_validation = X_test
    y_real = Y_test

yhat = model.predict(X_validation)

y_real = dtm.unnormalize(y_real,
                         [MinValues['GBPUSD_CLOSE'],
                          MaxValues['GBPUSD_CLOSE']])
yhat = dtm.unnormalize(yhat,
                       [MinValues['GBPUSD_CLOSE'],
                        MaxValues['GBPUSD_CLOSE']])

plt.figure(figsize = (15,10))
plt.plot(y_real[:])
plt.plot(yhat[:])
plt.legend(['real','predito'])

plt.figure(figsize = (15,10))
plt.plot(df['GBPUSD_CLOSE'][-len(yhat):].reset_index(drop=True))
plt.plot(yhat[:]);plt.legend(['real','predito'])

# writing model results into files
```

```
                    rs.writeData(params.file_path,directory,history,
                                 params,coinPair = params.name)
                    params.file_path = rs.save_model(model,params)
                    print(resource.getrusage(resource.RUSAGE_SELF).ru_maxrss)

                    del model;del history;
                    gc.collect()

mail_sender(NOME_DIR, 'GRFICOS PLOTADOS E SALVOS')
# EOF
```