**Estratégia de Atendimento à População no Combate à Covid-19**

**Simulador de Cenários com Inteligência Artificial e Dinâmica de Sistemas**

Guttenberg Ferreira Passos

Ilan Chamovitz

Suellen Teixeira Zavadzki de Pauli

Paulo Edy Nakamura

Leonardo Grandinetti Chaves

O presente artigo tem como objetivo contribuir para uma estratégia de atendimento à população no combate à Covid-19, através de um simulador de cenários com Inteligência Artificial e Dinâmica de Sistemas.

Em fevereiro de 2021 foram anunciadas no Brasil mudanças relevantes no combate à Covid-19, especialmente quanto à vacinação. Aborda-se nesse artigo uma série de previsões para auxiliar na estratégia de atendimento à população no combate a esta doença, é uma boa oportunidade para entender melhor os fatores "invisíveis" que afetam a contaminação das pessoas.

Para ajudar a modelar esses fatores "invisíveis", partiu-se da obtenção de dados de distanciamento social, esses dados foram passados como entrada em um sistema de Inteligência Artificial e foram executadas previsões em Deep Learning (LSTM). A previsão da variável distanciamento social foi exportada para calibrar o modelo em Systems Dynamics, com o software Stella, e as simulações foram feitas com um modelo de Covid-19. Como trabalho futuro, o resultado dessas simulações serão plotados em um mapa de um telefone celular através de geoprocessamento.

Todos os modelos foram adaptados das aulas do curso de Modelagem Preditiva de Séries Temporais que podem ser encontradas na timeline da Comunidade da Data Science Academy no portal: [www.datascienceacademy.com.br](http://www.datascienceacademy.com.br/).

Séries Temporais

Uma série temporal é um conjunto sequencial de pontos de dados, medido tipicamente em tempos sucessivos.É matematicamente definido como um conjunto de vetores x (t), t = 0,1,2, ... onde t representa o tempo decorrido. A variável x(t) é tratada como uma variável aleatória.

Uma variável aleatória é uma variável quantitativa, cujo resultado (valor) depende de fatores aleatórios.

As medições realizadas durante um evento em uma série temporal são organizadas em uma ordem cronológica adequada.Uma série temporal contendo registros de uma única variável é denominada como univariada e mais de uma variável como multivariada.

As séries temporais adicionam uma dependência explícita da ordem entre as observações: uma dimensão temporal. Essa dimensão adicional é uma restrição e uma estrutura que fornece uma fonte de informações adicionais. E muito, muito valiosa.

A análise de séries temporais envolve o desenvolvimento de modelos que melhor capturam ou descrevem uma série temporal observada para entender as causas. Este campo de estudo busca o "porquê" por trás de um conjunto de dados de séries temporais.

A qualidade de um modelo descritivo é determinada por quão bem ele descreve todos os dados disponíveis e a interpretação que fornece para melhor informar o domínio do problema.

O objetivo principal da análise de séries temporais é desenvolver modelos matemáticos que forneçam descrições plausíveis a partir de dados de amostra.

Fazer previsões sobre o futuro é chamado de extrapolação no tratamento estatístico clássico de dados de séries temporais. Os campos mais modernos se concentram no tópico e se referem a ele como previsão de séries temporais. A previsão envolve ajustar os modelos aos dados históricos e usá-los para prever observações futuras.

Uma distinção importante na previsão é que o futuro está completamente indisponível e só deve ser estimado a partir do que já aconteceu. O objetivo da análise de séries temporais é geralmente duplo: entender ou modelar os mecanismos estocásticos que dão origem a uma série observada e prever os valores futuros de uma série com base no histórico dessa série. Isso é o que chamamos de Modelagem Preditiva.

Modelagens Preditivas não lineares com séries temporais.

* Para modelos preditivos não lineares usa-se Métodos Estatísticos, Inteligência Artificial, Machine Learning e Deep Learning, tais como os seguintes algoritmos:
* Naive;
* ExponentialSmoothing;
* ARMA - Autoregressive Moving Average;
* ARIMA - Autoregressive Integrated Moving Average;
* SARIMAX – Seasonal Autoregressive Integrated Moving Average;
* DeepAR–GluonTS - Probabilistic Time Series Modeling;
* MLP - MultilayerPerceptron;
* LSTM - Long short-term memory.

Algoritmo Naive

É o modelo preditivo mais simples que se pode criar. Baseia-se em uma técnica de estimativa na qual os dados reais do último período são usados como previsão desse período, sem ajustá-los ou tentar estabelecer fatores causais. É usado apenas para comparação com as previsões geradas pelas melhores técnicas (sofisticadas).

Naivesignifica ingênuo, portanto, não há técnica avançada aqui é apenas usadocomo ponto de partida, uma referência para os demais modelos. Qualquer modelo mais avançado deve apresentar resultados superiores ao AlgoritmoNaive.

Algoritmo ExponentialSmoothing

Smoothing (Suavização ou Alisamento) em séries temporais é um conjunto de métodos para suavizar séries temporais eliminando "saltos". Existem várias maneiras de fazer isso. Talvez o mais fácil seja calcular a média móvel simples (SimpleMovingAverage).

A suavização é basicamente uma técnica usada para ver a tendência de longo prazo nos dados, diminuindo os efeitos dos componentes periódicos / sazonais dos dados. Basicamente, usa-se a suavização quando se quer remover as flutuações nos dados e focar apenas em preservar as tendências de longo prazo.

O objetivo de suavizar é remover o ruído e expor melhor o sinal dos processos. As médias móveis são um tipo simples e comum de suavização usado na análise de séries temporais e na previsão de séries temporais. O cálculo de uma média móvel envolve a criação de uma nova série em que os valores são compostos da média de observações brutas na série temporal original.

A suavização da média móvel (MovingAverageSmoothing) é uma técnica eficaz na previsão de séries temporais também, ou seja, pode ser usado para preparação de dados, engenharia de recursos e até diretamente para fazer previsões. Uma média móvel requer que você especifique um tamanho de janela chamado largura da janela. Isso define o número de observações brutas usadas para calcular o valor da média móvel.

A parte "móvel" na média móvel refere-se ao fato de que a janela definida pela largura da janela é deslizada ao longo da série temporal para calcular os valores médios na nova série.

Algoritmo ARMA

Auto RegressiveMovingAverage - ARMA é um modelo de média móvel auto-regressiva. É simplesmente a fusão entre os modelos AR (p) e MA (q). O modelo AR (p) tenta explicar o momento e os efeitos médios da reversão frequentemente observados nos mercados (efeitos dos participantes do mercado). O modelo MA (q) tenta capturar os efeitos de choque observados em termos de ruído branco. Estes efeitos de choque podem ser considerados eventos inesperados que afetam o processo de observação, p, como ganhos repentinos, guerras, ataques, etc.

O modelo ARMA tenta capturar esses dois aspectos ao modelar séries temporais, não leva em consideração o agrupamento de volatilidade, um fenômeno empírico essencial de muitas séries temporais financeiras.

O Modelo ARMA(1,1) é representado como: x(t) = ax(t-1) + be(t-1) + e(t), onde e(t) é o ruído branco com E [e(t)] = 0.

Um modelo ARMA geralmente requer menos parâmetros que um modelo AR (p) ou um modelo MA (q) individual.

Algoritmo ARIMA

AutoregressiveIntegratedMovingAverage - ARIMAé um modelo de média móvel integrada auto-regressiva, uma generalização de um modelo ARMA.

Ambos os modelos são ajustados a dados de séries temporais para melhor entender os dados ou para prever pontos futuros na série (previsão). Os modelos ARIMA são aplicados em alguns casos em que os dados mostram evidências de não estacionariedade, onde uma etapa inicial de diferenciação (correspondente à parte "integrada" do modelo) pode ser aplicada uma ou mais vezes para eliminar a não estacionariedade.

A parte AR do ARIMA indica que a variável de interesse em evolução é regredida com seus próprios valores defasados (isto é, anteriores). O I (para "integrado") indica que os valores dos dados foram substituídos pela diferença entre seus valores e os valores anteriores (e esse processo de diferenciação pode ter sido executado mais de uma vez). O objetivo de cada um desses recursos é fazer com que o modelo ajuste os dados da melhor maneira possível. A parte MA indica que o erro de regressão é na verdade uma combinação linear de termos de erro cujos valores ocorreram contemporaneamente e em vários momentos no passado.

Modelos ARIMA não sazonais são geralmente designados ARIMA(p, d, q), em que os parâmetros p, d e q são números inteiros não negativos, p é a ordem (número de intervalos de tempo) do modelo autoregressivo, d é o grau de diferenciação (o número de vezes que os dados tiveram valores passados subtraídos) e q é a ordem do modelo de média móvel.

Algoritmo SARIMAX

SeasonalAutoregressiveIntegratedMovingAverage- SARIMAX é um modelo sazonal de média móvel integrada auto-regressivacom os parâmetros (p, d, q) (P, D, Q) m, em que m refere-se ao número de períodos em cada sazonalidade e os maiúsculos P, D, Q referem-se ao autorregressivo, diferenciado, e termos da média móvel da parte sazonal do modelo ARIMA.

Algoritmo DeepAR-GluonTS

O DeepARfoi lançado pela Amazon, sendo integrado ao SageMaker. Seu objetivo é prever a cada etapa o seguinte (horizonte = 1). Isso significa que a rede deve receber na entrada a observação anterior (no atraso = 1) z\_t-1, juntamente com um conjunto de covariáveis opcionais x\_i. As informações são propagadas para a camada oculta e até a função de verossimilhança (que é uma função de pontuação usada no nível de uma função de perda). A função de probabilidade pode ser binomial gaussiana ou negativa. O erro é calculado usando a parametrização atual da probabilidade. Facilmente, isso é representado por mu e sigma no caso de uma probabilidade gaussiana. Isso significa que, durante a execução do backpropagation, estamos ajustando os parâmetros de rede (pesos w) que alteram a parametrização de todos os exemplos.

O DeepAR pode ser implementado com o Gluon Time Series - GluonTS, uma biblioteca para modelagem de séries temporais baseada em Deep Learning, simplifica o desenvolvimento e a experimentação de modelos de séries temporais para tarefas comuns, como previsão ou detecção de anomalias. Ele fornece todos os componentes e ferramentas necessários que os Cientistas de Dados precisam para criar rapidamente novos modelos, executar e analisar experimentos com eficiência e avaliar a precisão do modelo.

Algoritmo MLP

Uma rede padrão MultilayerPerceptron- MLPé uma rede neural recorrente - RNNprojetada para problemas seqüenciais,pode ser pensada como a adição de loops à arquitetura. Por exemplo, em uma dada camada, cada neurônio pode passar seu sinal para frente (feed-forward) e também para o lado.

Uma Rede Neural Recorrente é basicamente uma rede neural que pode ser usada quando seus dados são tratados como uma sequência, onde a ordem particular dos pontos de dados é importante e esta sequência pode ser de comprimento arbitrário.

O exemplo mais claro é talvez uma série temporal de números, onde a tarefa é prever o próximo valor de acordo com valores anteriores. A entrada para a RNN em cada passo de tempo é o valor atual, bem como um vetor de estado que representa o que a rede "viu" no tempo - etapas anteriores. Este estado-vetor é a memória codificada da RNN, inicialmente definida como zero.

Algoritmo LSTM

A rede Long Short-TermMemory- LSTMé uma rede neural recorrente treinada usando BackpropagationThrough Time e supera o problema da dissipação do gradiente.Como tal, o modelo pode ser usado para criar grandes redes recorrentes que, por sua vez, podem ser usadas para resolver problemas de sequência difíceis no aprendizado de máquina e obter resultados de última geração.

Em vez de neurônios, as redes LSTM possuem blocos de memória conectados através de camadas.Um bloco possui componentes que o tornam mais inteligente que um neurônio clássico e uma memória para sequências recentes. Um bloco contém portas que gerenciam o estado e a saída do bloco. Um bloco opera sobre uma sequência de entrada e cada porta dentro de um bloco usa as unidades de ativação sigmóide para controlar se são acionadas ou não, condicionando a mudança de estado e a adição de informações que fluem através do bloco.

Neurônio Biológico

O Deep Learning Book Brasil é uma iniciativa da [Data Science Academy](https://www.datascienceacademy.com.br/), com o objetivo de ajudar a difundir o Deep Learning, uma das tecnologias mais revolucionárias do nosso tempo usada na construção de aplicações de Inteligência Artificial.

De acordo o com o Deep Learning Book, o neurônio biológico é uma célula, que pode ser dividida em três seções: o corpo da célula, os dendritos e o axônio, cada uma com funções específicas, porém complementares.



Fonte: Data Science Academy- DSA

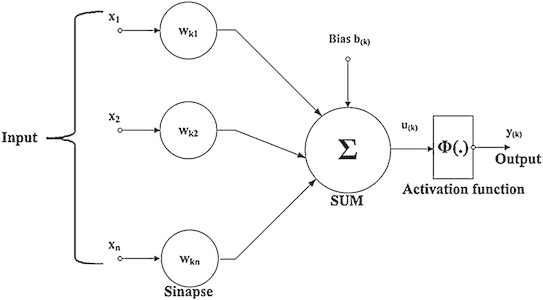
Neurônio Artificial

Um neurônio artificial representa a base de uma Rede Neural Artificial (RNA), um modelo da neuroinformática orientado nas redes neurais biológicas.



Fonte: Data Science Academy - DSA

O conhecimento de uma RNA está codificado na estrutura da rede,onde se destacam as conexões (sinapses)entre as unidades (neurônios) que a compõe.



Fonte: Data Science Academy - DSA

No aprendizado de máquina, o Perceptron é um algoritmo de aprendizado supervisionado de classificadores binários. Um classificador binário é uma função que pode decidir se uma entrada, representada por um vetor de números, pertence ou não a alguma classe específica. É um tipo de classificador linear, ou seja, um algoritmo de classificação que faz suas previsões com base em uma função de preditor linear combinando um conjunto de pesos com o vetor de características.

Redes Neurais

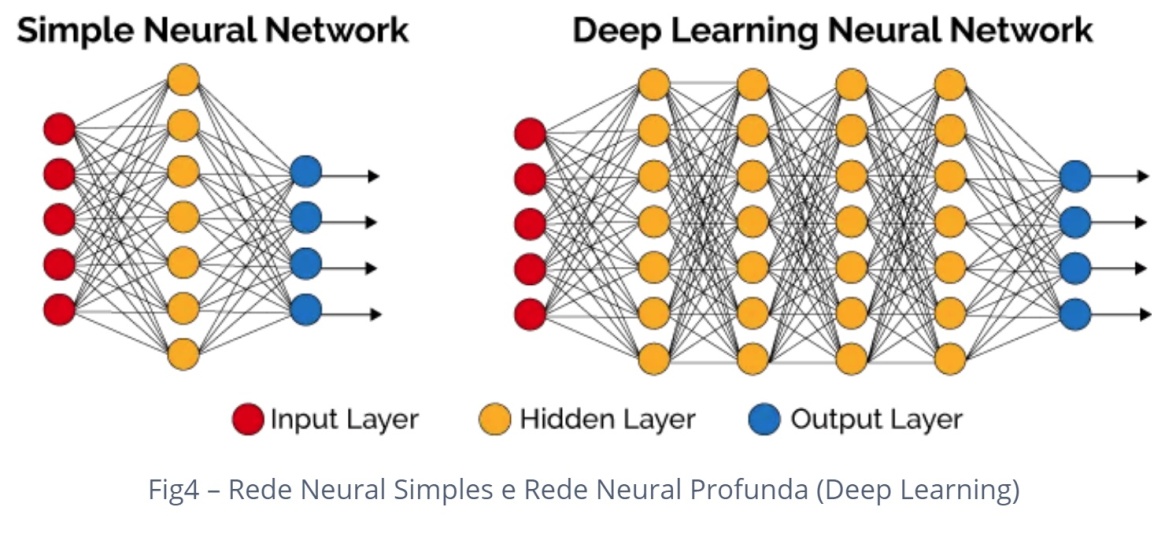
Redes neurais são sistemas de computação com nós interconectados que funcionam como os neurônios do cérebro humano. Usando algoritmos, elas podem reconhecer padrões escondidos e correlações em dados brutos, agrupá-los e classificá-los, e com o tempo aprender e melhorar continuamente.

O Instituto Asimov <https://www.asimovinstitute.org/neural-network-zoo/> publicou uma folha de dicas contendo várias arquiteturas de rede neural, nos concentraremos nas arquiteturas abaixo com foco em Perceptron(P), FeedFoward (FF), Recurrent Neural Network (RNN) e Long Short TermMemory (LSTM):



Fonte: THE ASIMOV INSTITUTE

Deep Learning é uma das bases da Inteligência Artificial (IA), um tipo de aprendizado de máquina (MachineLearning)  que treina computadores para realizar tarefas como seres humanos, o que inclui reconhecimento de fala, identificação de imagem e previsões, aprendendo com o tempo. Podemos dizer que é uma Rede Neural com várias camadas ocultas:



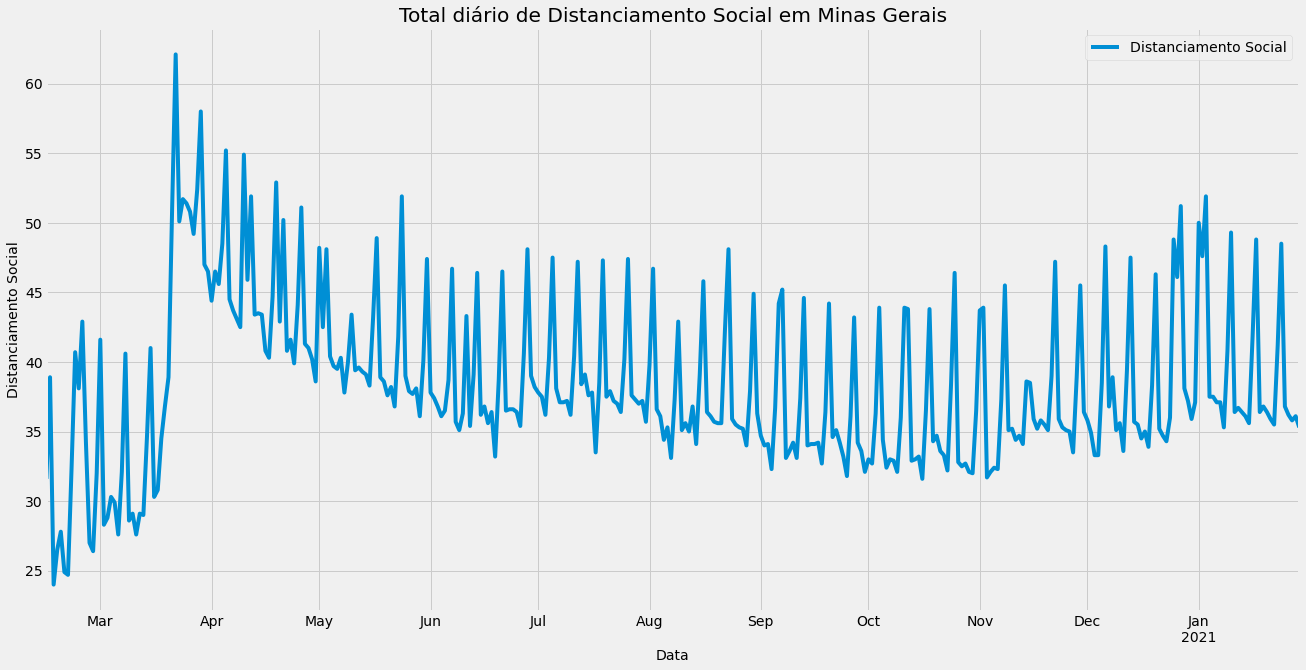
**Modelo Base**

**Definição do Problema de Negócio**

Previsão do distanciamento social em Minas Gerais.

**Conjunto de Dados**

Usaremos conjuntos de dados que mostram o distanciamento social em Minas Gerais. Os dados têm registros de 15/02/2020 a 29/01/2021.

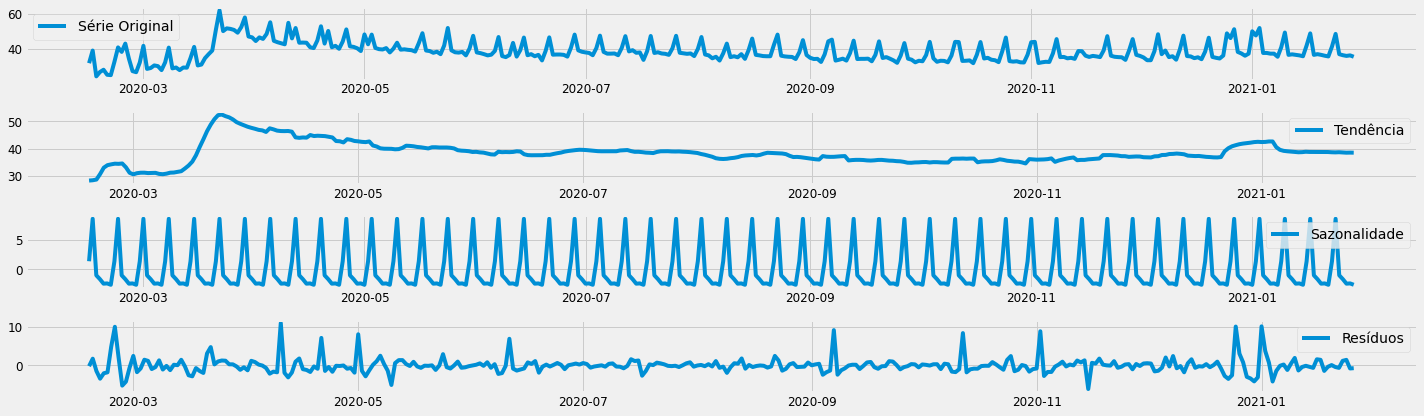


**Análise Exploratória dos Dados**

**Decomposição da Série Temporal**

Pode-se usar os modelos estatísticos para realizar uma decomposição dessa série cronológica. A decomposição de séries temporais é uma tarefa estatística que desconstrói uma série temporal em vários componentes, cada um representando uma das categorias de padrões. Com os modelos de estatísticas, poderemos ver a tendência, os componentes sazonais e residuais de nossos dados. Pode-se fazer uma decomposição clássica de uma série temporal, considerando a série como uma combinação aditiva ou multiplicativa do nível base, tendência, índice sazonal e residual.

Segue abaixo a decomposição da série temporal:



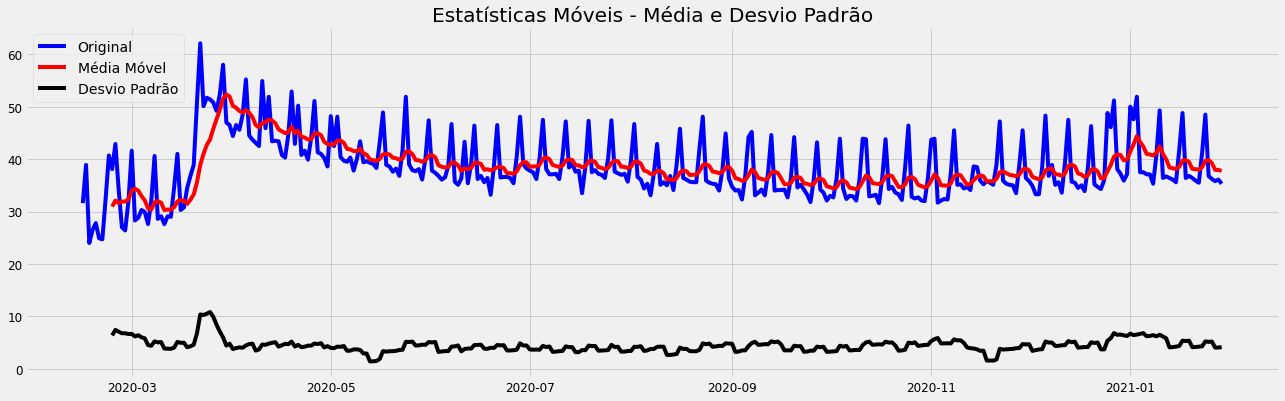
O gráfico acima mostra 2 componentes da série: Sazonalidade e Tendência.

* Sazonalidade - o fenômeno se repete em períodos fixos.
* Tendência - ao longo do tempo, a série segue uma tendência de crescimento.

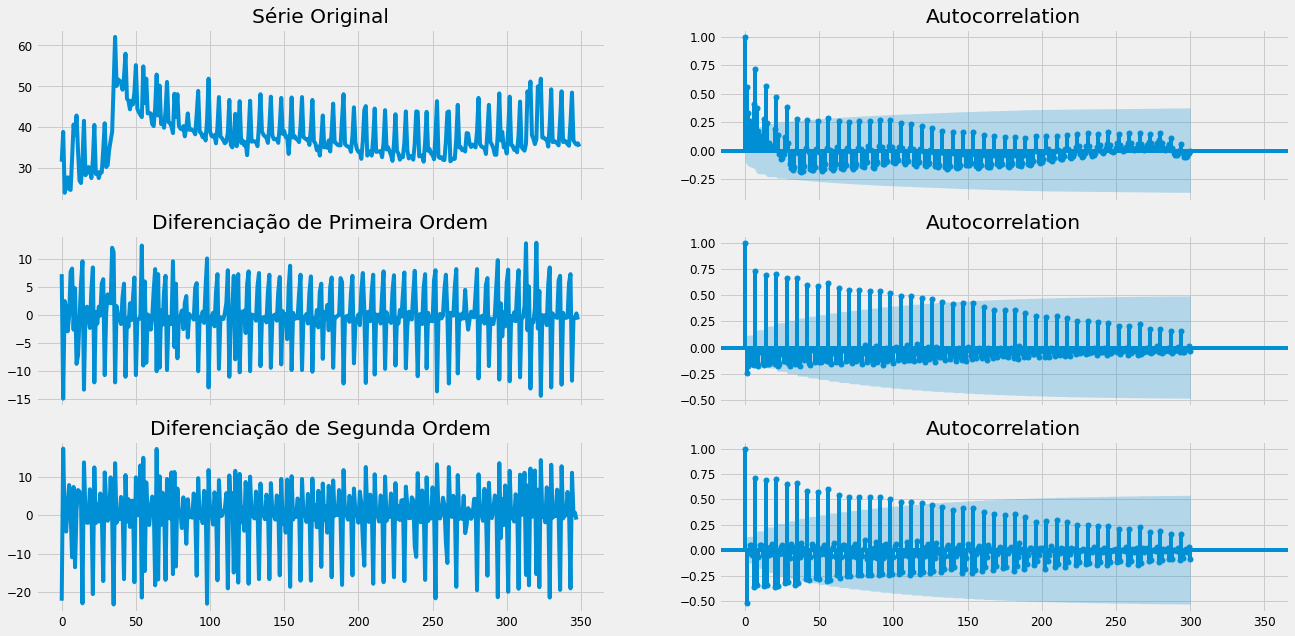
**Estacionaridade da série.**

A estacionariedade é um importante conceito na modelagem de séries temporais e é caracterizada por uma variável que se comporta de forma aleatória ao longo do tempo ao redor de uma média constante. Uma série temporal é considerada estacionária se suas propriedades estatísticas, como média e variância, permanecerem constantes ao longo do tempo. Intuitivamente, podemos dizer que, se uma série temporal tem um comportamento específico ao longo do tempo, há uma probabilidade muito alta de que ela siga o mesmo no futuro.

Basicamente, séries temporais que possuem tendência e/ou sazonalidade não são estacionárias e é necessário o uso de técnicas adequadas a tal situação.



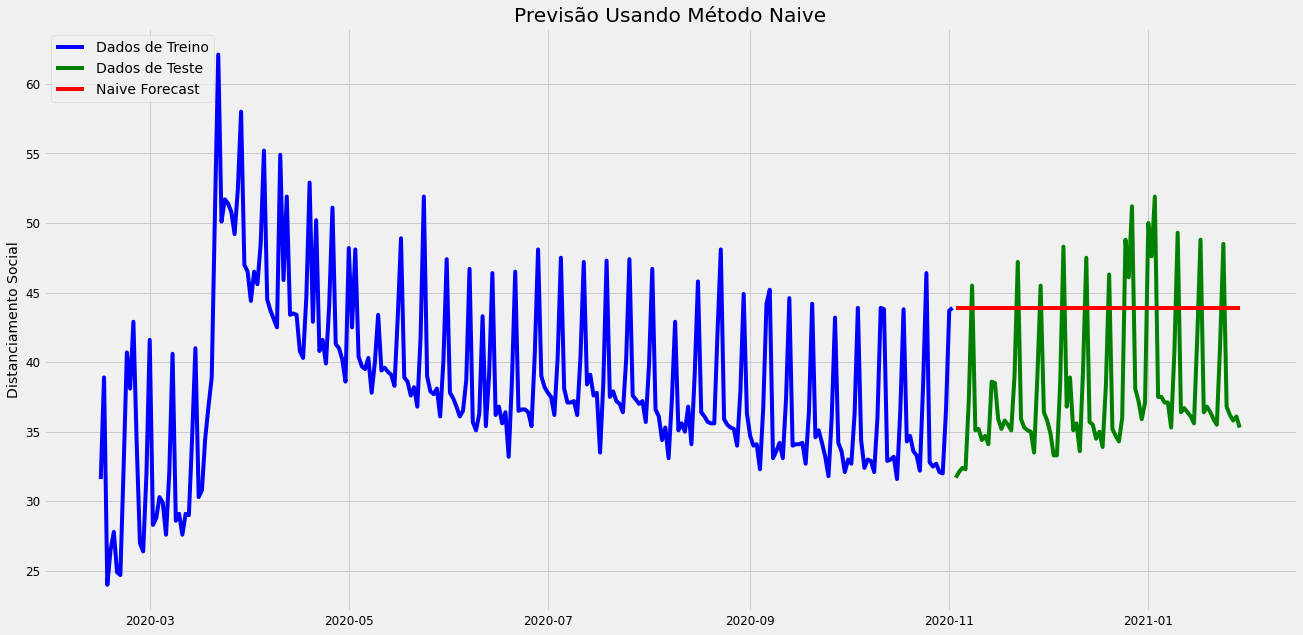
O gráfico ACF permite a avaliação da diferenciação mínima necessária para obter uma série estacionária (Parâmetro d para o Modelo ARIMA):



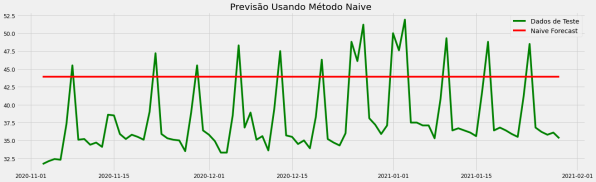
Para a execução dos modelos utilizando **Métodos Estatísticos** os dados foram separados em:

* 262 registros de treino e
* 88 registros de validação.

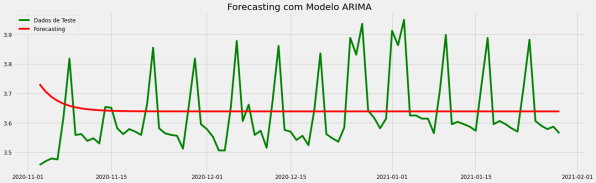
**Modelo 11 – Previsões Método Naive**



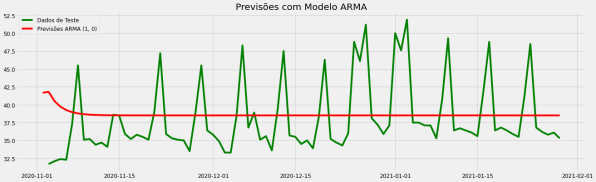
**11 – Método Naive**



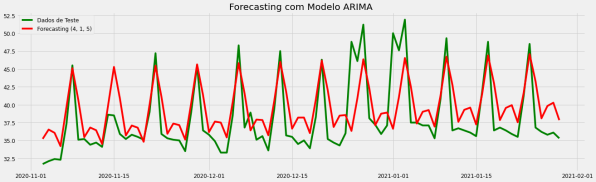
**13 - Forecasting–ARIMALOG (1, 0, 1)**



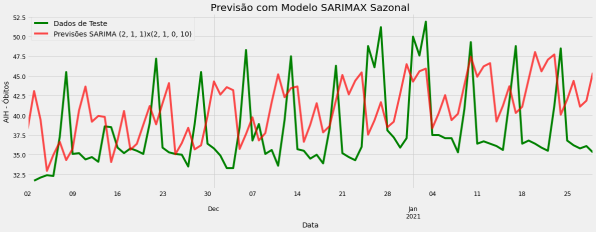
**15 –ARMA (1, 0)**



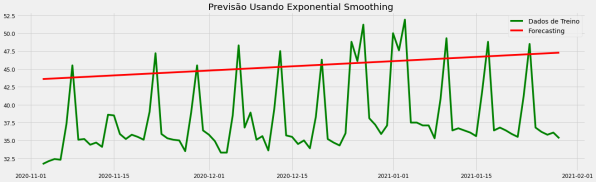
**16 – Forecasting - ARIMA (4, 1, 5)**



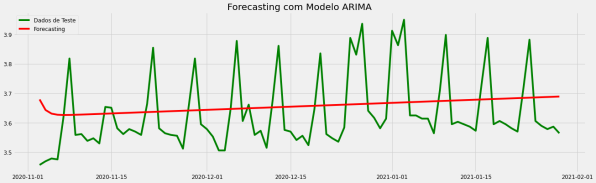
**17 – SARIMAX(2,1,1)(2,1,0,10)**



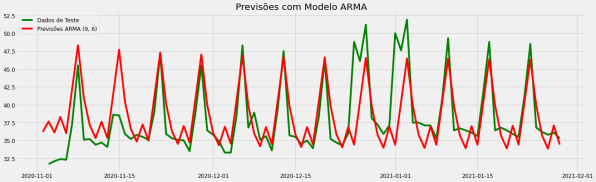
**12 – ExponentialSmoothing**



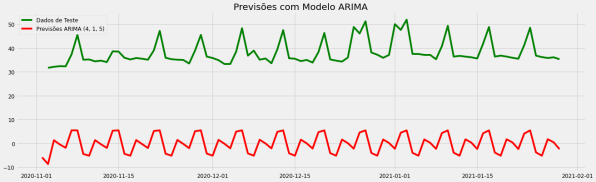
**14 - Forecasting– ARIMALOG (1, 1, 1)**



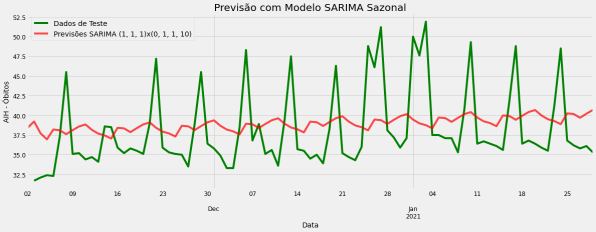
**15 – ARMA (9, 6)**



**16 - Predict-ARIMA (4, 1, 5)**



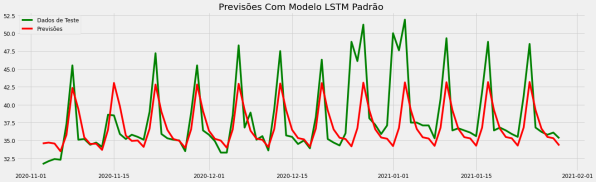
**18 – SARIMAX(1,1,1)(0,1,1,10)**



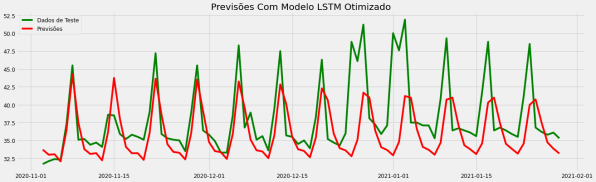
**Modelos utilizando**

**Inteligência Artificial - IA**

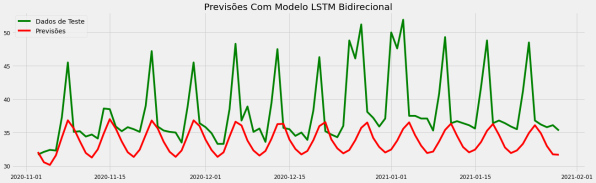
**22 – LSTM - IA (3 repetições)**



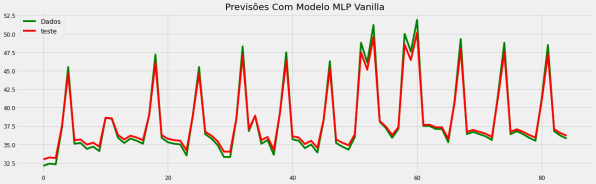
**23 – LSTM otimizado - IA**



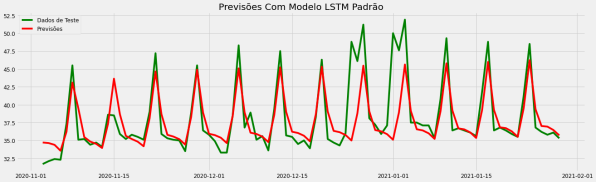
**25 – LSTM Bidirecional - IA**



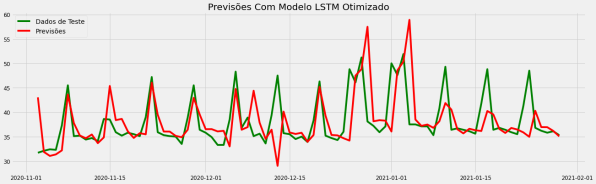
**RNN01 – MLP Vanilla - IA**



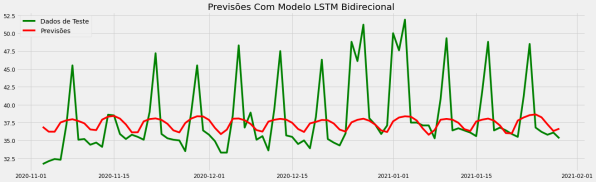
**22 – LSTM - IA (5 repetições)**



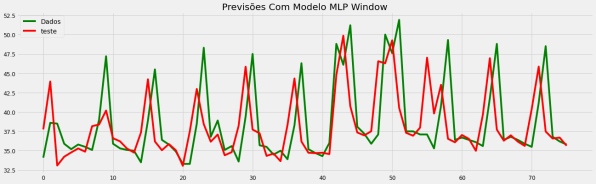
**24 – Stacked LSTM - IA**



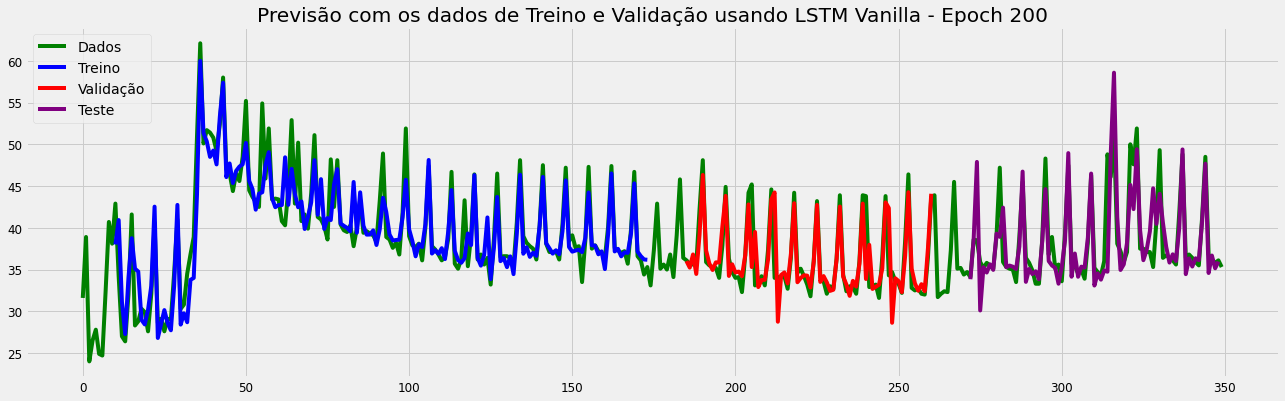
**25 – LSTM - IA - Grid Search**



**RNN01 – MLP Window - IA**

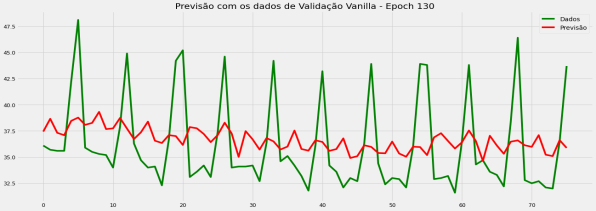


**Previsões utilizando Inteligência Artificial - IA - Deep Learning**



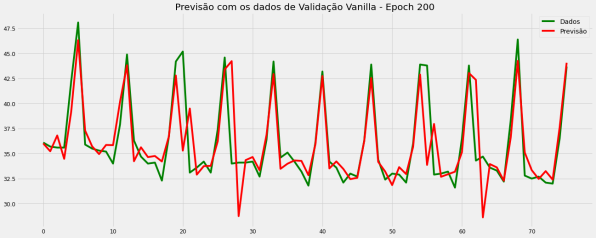
**RNN02 – LSTM Vanilla IA – Validação**

**Epoch = 130 - evita o Overfitting**



**RNN02 – LSTM Vanilla IA – Validação**

**Epoch = 200**



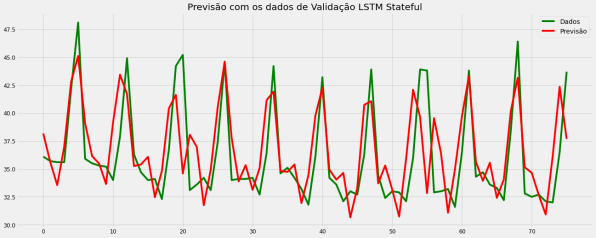
**RNN02 – LSTM Window IA– Validação**



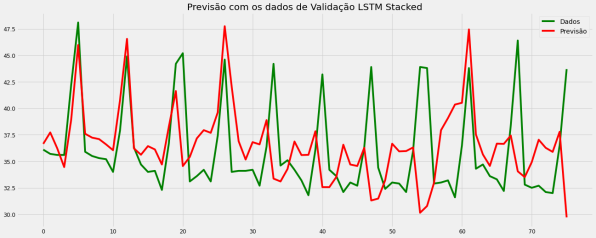
**RNN02– LSTM Time Steps IA Validação**



**RNN02 – LSTM Stateful IA – Validação**



**RNN02 – LSTM Stacked IA – Validação**



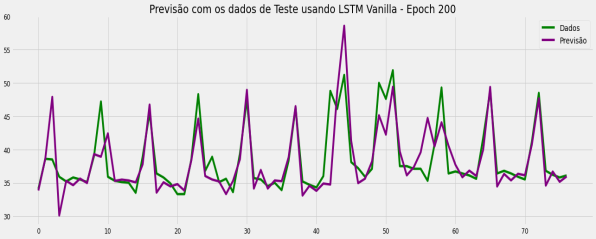
**RNN02 – LSTM Vanilla IA – Teste**

**Epoch = 130 - evita o Overfitting**

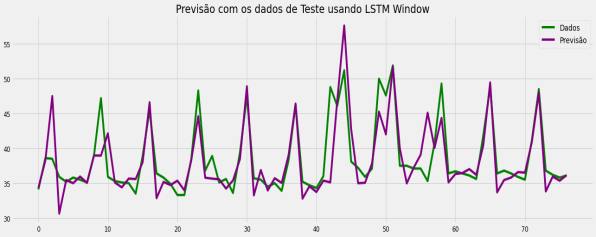


**RNN02 – LSTM Vanilla IA – Teste**

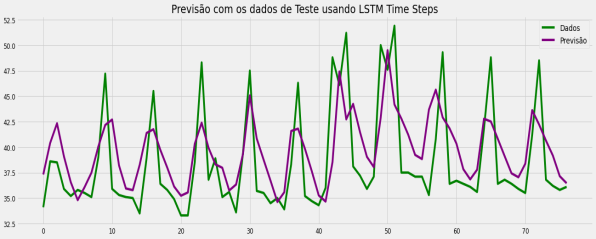
**Epoch = 200**



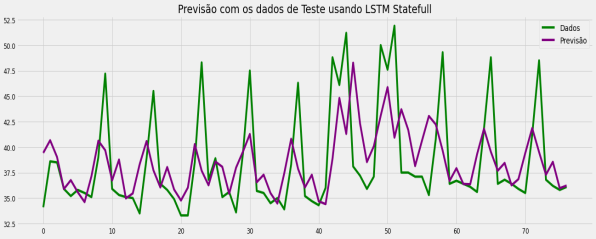
**RNN02 – LSTM Window IA – Teste**



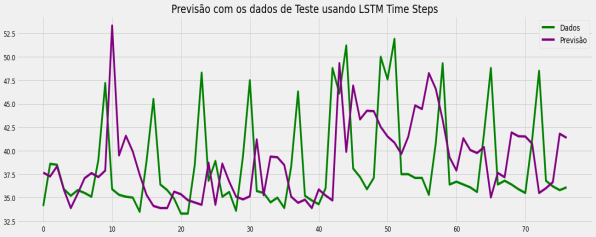
**RNN02 – LSTM TimeSteps IA – Teste**



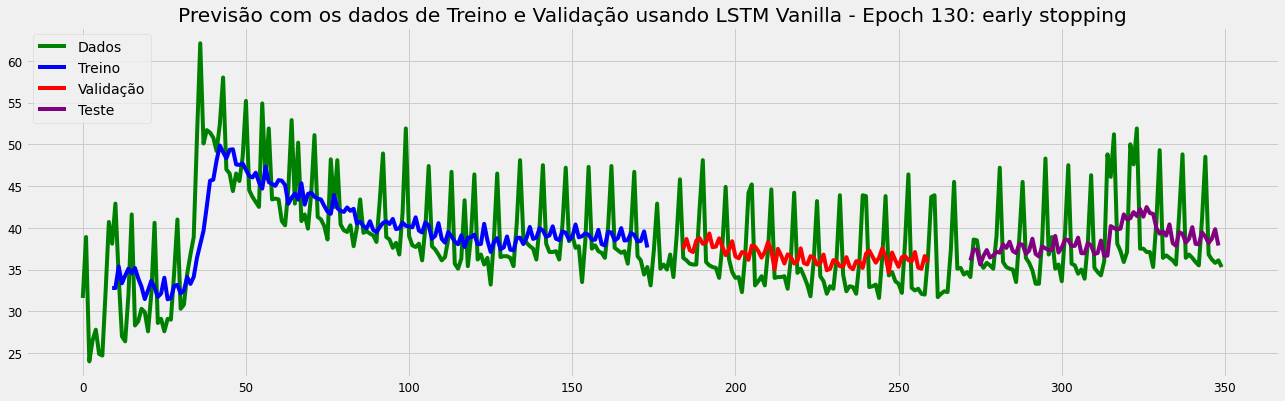
**RNN02 – LSTM Stateful IA – Teste**



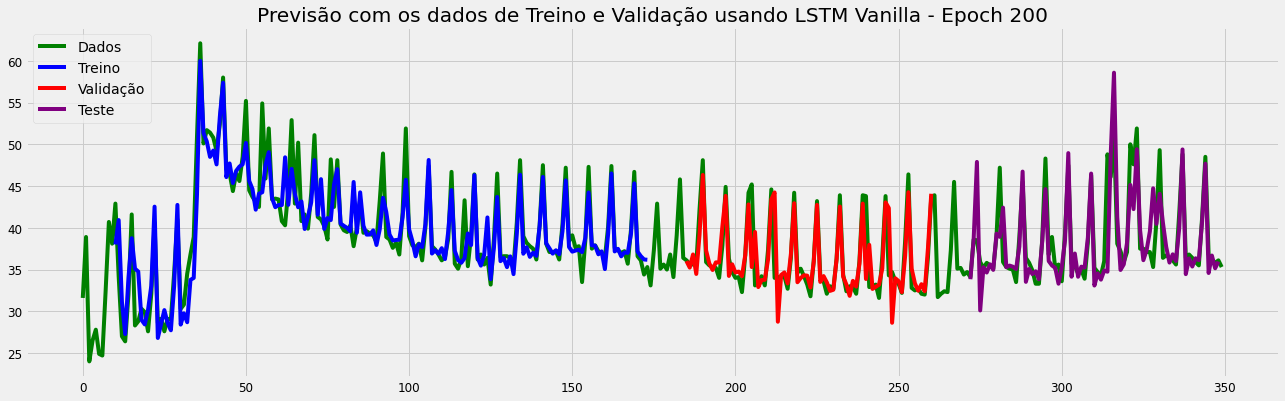
**RNN02 – LSTM Stacked IA – Teste**



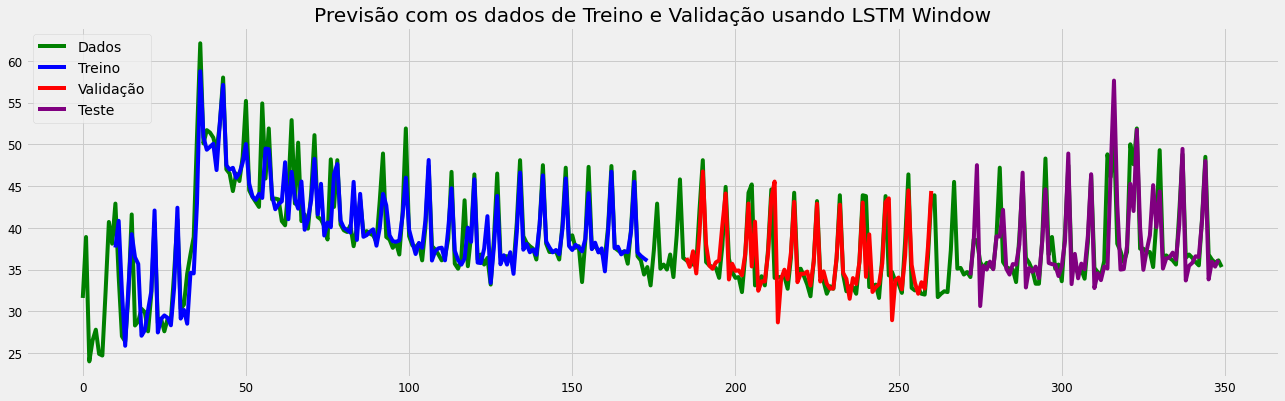
**Modelo RNN02 – LSTM Vanilla IA – Epoch = 130 – evita o Overfitting**



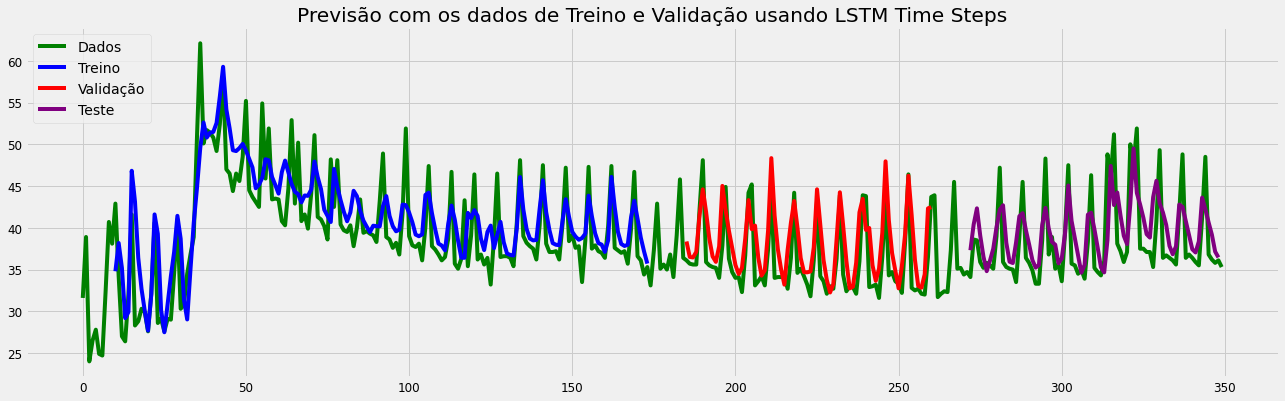
**Modelo RNN02 – LSTM Vanilla IA – Epoch = 200**



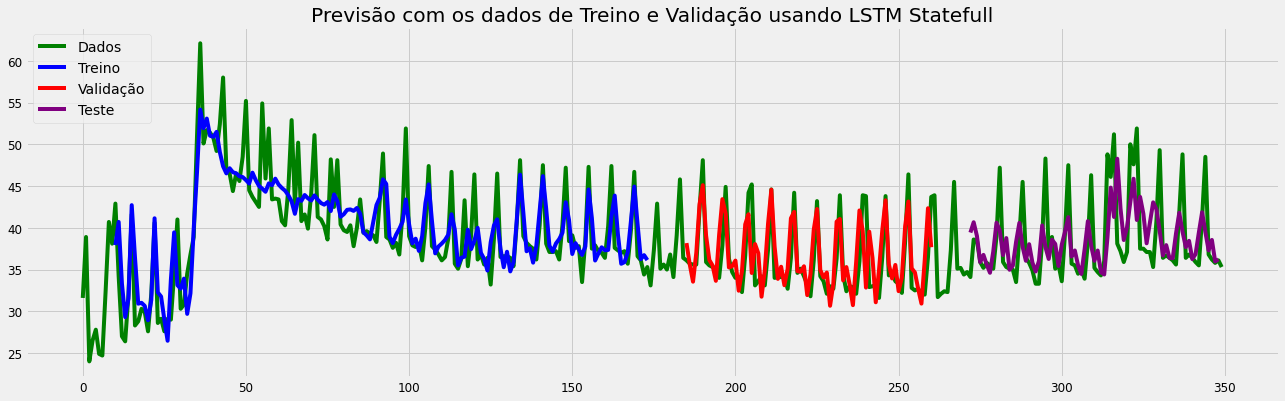
**ModeloRNN02 – LSTM Window IA**



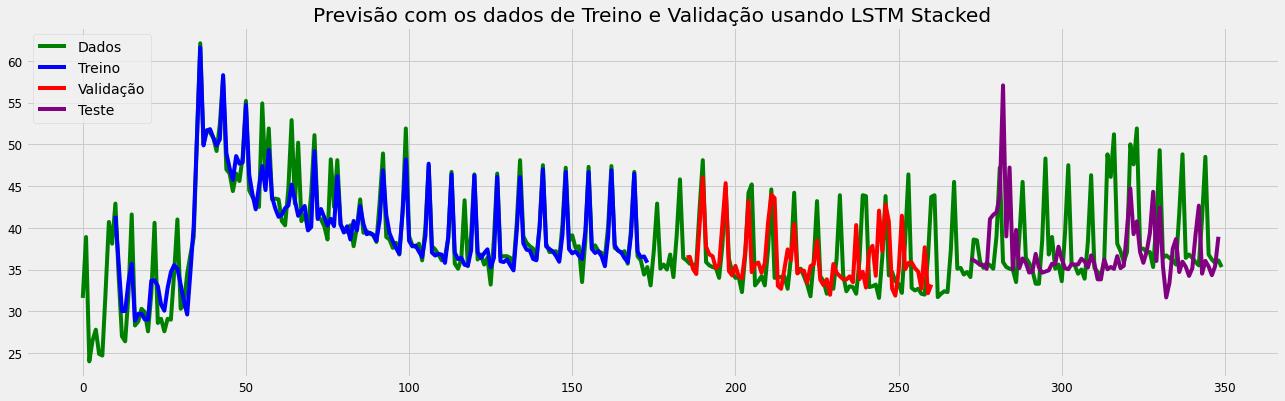
**ModeloRNN02– LSTM Time Steps**



**ModeloRNN02– LSTM Stateful**

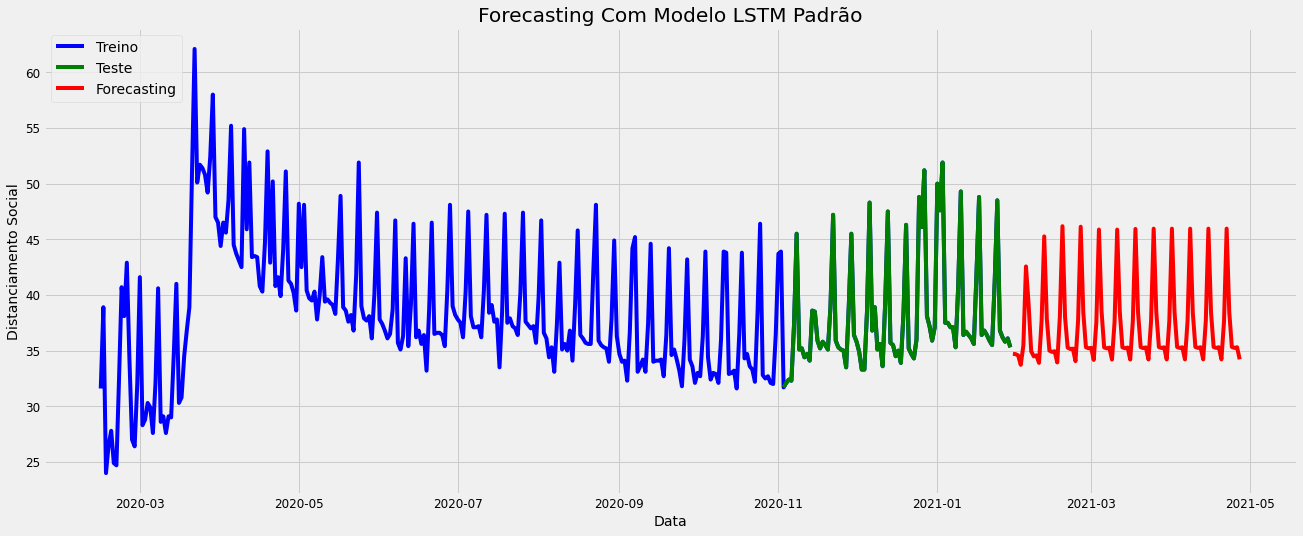


**Modelo RNN02– LSTM Stacked**

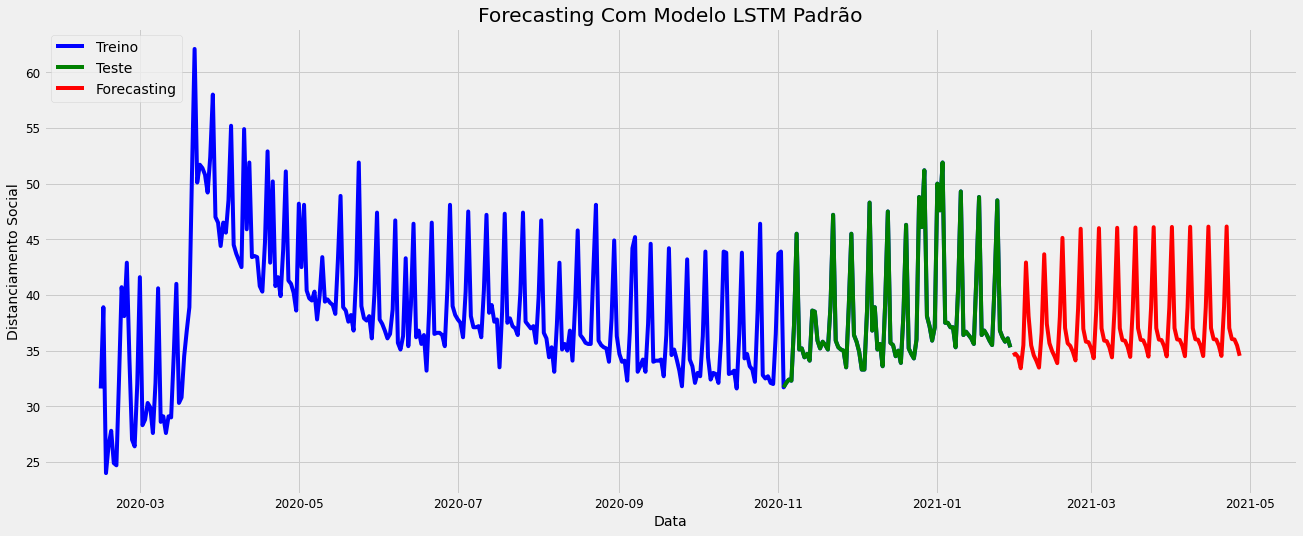


**Forecasting utilizando Inteligência Artificial - IA**

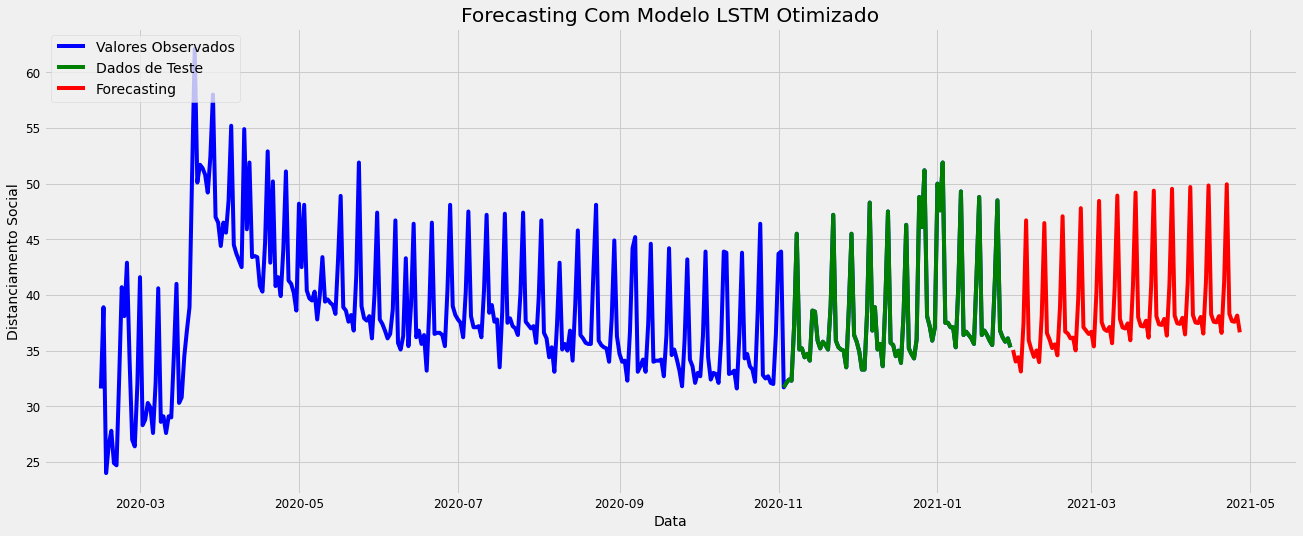
**Modelo 22 – LSTM - IA (3 repetições)**



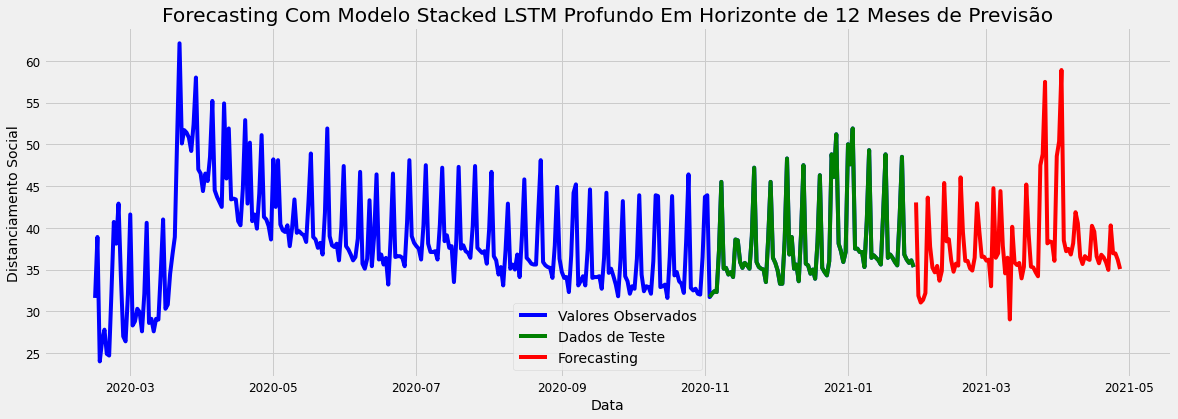
**Modelo 22 – LSTM - IA (5 repetições)**



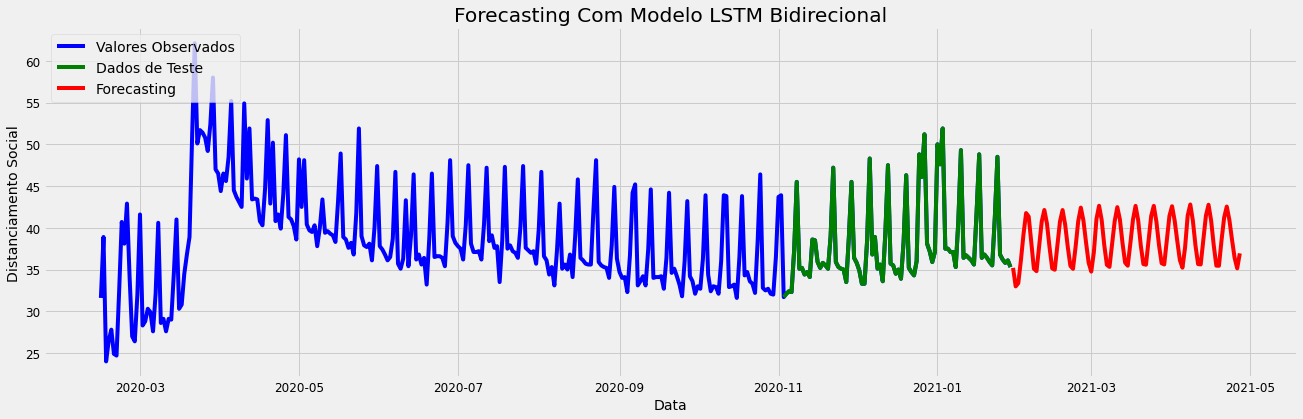
**Modelo 23 – LSTM otimizado - IA**



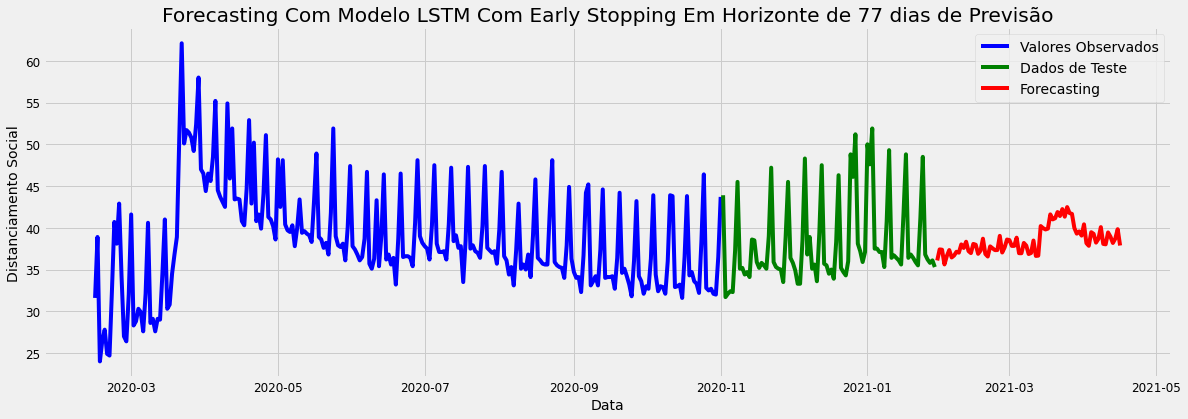
**Modelo 24 – Stacked LSTM - IA**



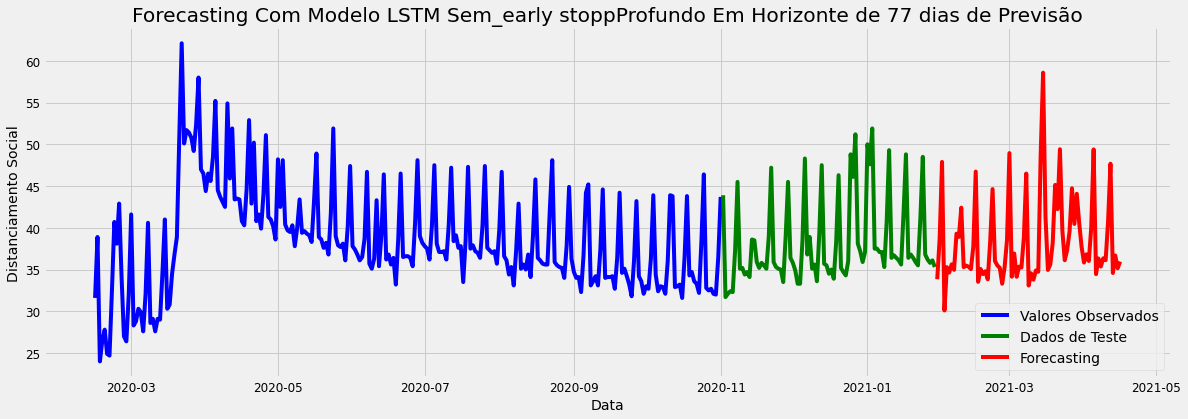
**Modelo 25 – Bidirecional LSTM - IA**



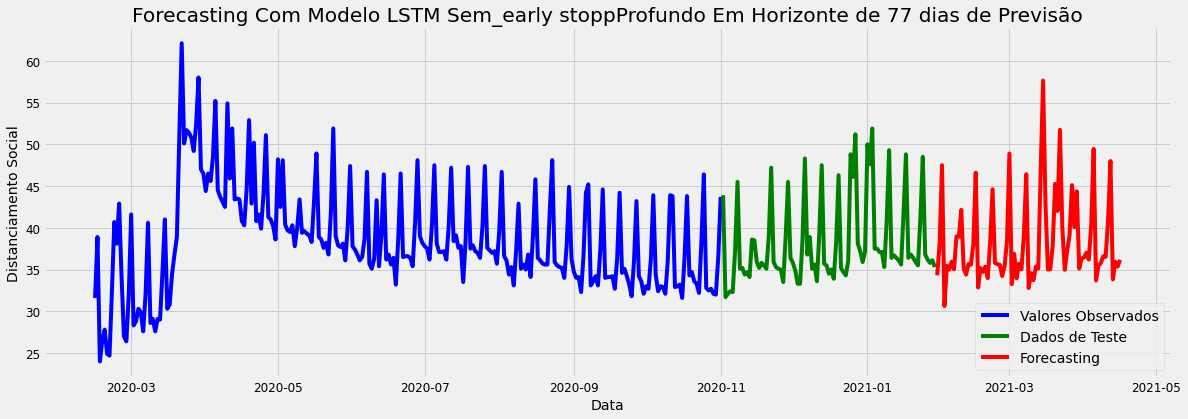
**Modelo RNN02 – LSTM Vanilla IA – Epoch = 130 – evita o Overfitting**



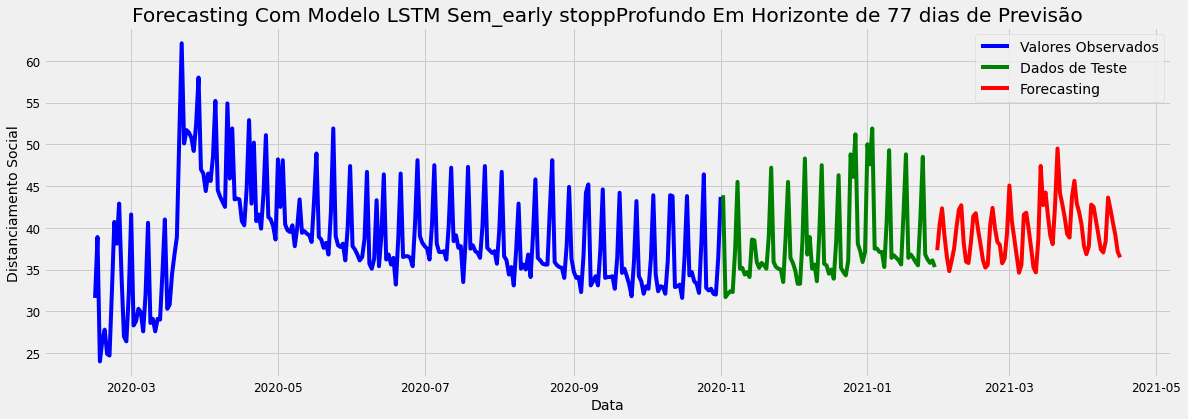
**Modelo RNN02 – LSTM Vanilla IA – Epoch = 200**



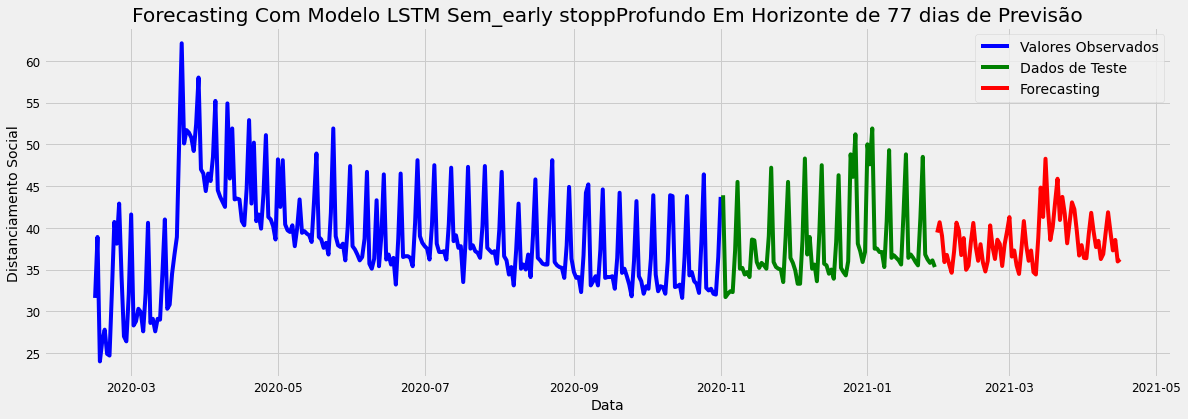
**ModeloRNN02 – LSTM Window IA**



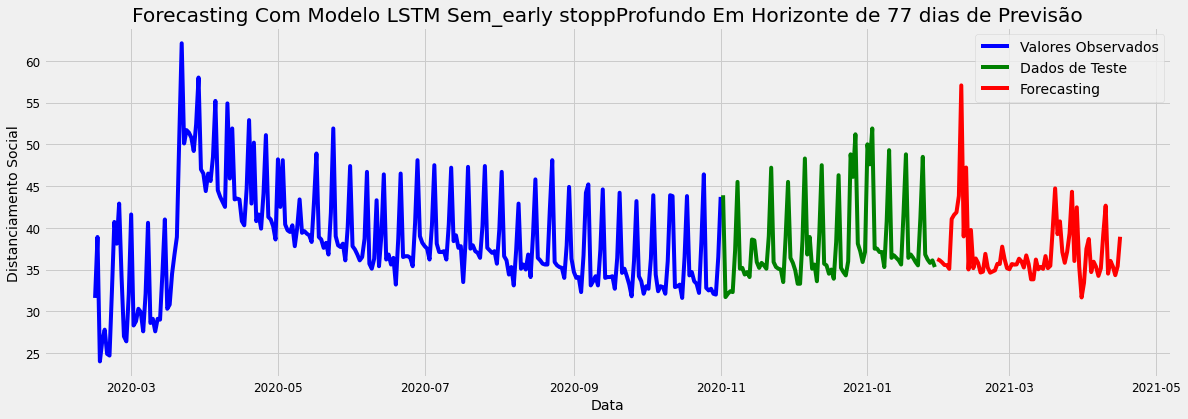
**ModeloRNN02– LSTM Time Steps**



**ModeloRNN02– LSTM Stateful**



**Modelo RNN02– LSTM Stacked**



**Resultados dos Modelos:**



**Arquiteturas dos Modelos utilizando IA**

**Modelo 22 – LSTM - IA (5 repetições)**

Número de repetições = 5

modelo\_lstm.add(LSTM(50, activation = 'relu', input\_shape = (n\_input, n\_features)))

modelo\_lstm.add(Dropout(0.10))

modelo\_lstm.add(Dense(100, activation = 'relu'))

modelo\_lstm.add(Dense(100, activation = 'relu'))

modelo\_lstm.add(Dense(1))

modelo\_lstm.compile(optimizer = 'adam', loss = 'mean\_squared\_error')

monitor = EarlyStopping(monitor='val\_loss', min\_delta=1e-3, patience=3, verbose=1, mode='auto')

modelo\_lstm.fit\_generator(generator, epochs = 200)

**Modelo 23 – LSTM otimizado - IA**

Número de repetições = 3

modelo\_lstm.add(LSTM(40, activation = 'tanh', return\_sequences = True, input\_shape = (n\_input, n\_features)))

modelo\_lstm.add(LSTM(40, activation = 'relu'))

modelo\_lstm.add(Dense(50, activation = 'relu'))

modelo\_lstm.add(Dense(50, activation = 'relu'))

modelo\_lstm.add(Dense(1))

adam = optimizers.Adam(lr = 0.001)

modelo\_lstm.compile(optimizer = adam, loss = 'mean\_squared\_error')

monitor = EarlyStopping(monitor='val\_loss', min\_delta=1e-3, patience=5, verbose=1, mode='auto')

modelo\_lstm.fit\_generator(generator, epochs = 100)

**Modelo 24 – Stacked LSTM - IA**

modelo\_lstm.add(LSTM(200, activation = 'relu', return\_sequences = True, input\_shape = (1, 1)))

modelo\_lstm.add(LSTM(100, activation = 'relu', return\_sequences = True))

modelo\_lstm.add(LSTM(50, activation = 'relu', return\_sequences = True))

modelo\_lstm.add(LSTM(25, activation = 'relu'))

modelo\_lstm.add(Dense(20, activation = 'relu'))

modelo\_lstm.add(Dense(10, activation = 'relu'))

modelo\_lstm.add(Dense(1))

modelo\_lstm.compile(optimizer = 'adam', loss = 'mean\_squared\_error')

modelo\_lstm.fit(X, y, epochs5000, verbose = 1)

**Arquiteturas dos Modelos utilizando IA Deep Learning**

**ModeloRNN01 – MLP Vanilla – IA**

model.add(Dense(8, input\_dim = look\_back, activation = 'relu'))

model.add(Dense(1))

model.compile(loss = 'mean\_squared\_error', optimizer = 'adam')

model.fit(trainX, trainY, epochs = 200, batch\_size = 2, verbose = 2)

**Modelo RNN01 – MLP Vanilla – IA**

model.add(LSTM(4, input\_shape = (1, look\_back)))

model.add(Dense(1))

model.compile(loss = 'mean\_squared\_error', optimizer = 'adam')

model.fit(trainX, trainY, epochs = 200, batch\_size = 1, verbose = 2)

**Modelo RNN02 – LSTM Vanilla – IA**

model.add(LSTM(4, input\_shape = (1, look\_back)))

model.add(Dense(1))

model.compile(loss = 'mean\_squared\_error', optimizer = 'adam')

model.fit(trainX, trainY, epochs = 200, batch\_size = 1, verbose = 2)

**Modelo RNN02 – LSTM Window – IA**

model.add(LSTM(4, input\_shape = (1, look\_back)))

model.add(Dense(1))

model.compile(loss = 'mean\_squared\_error', optimizer = 'adam')

model.fit(trainX, trainY, epochs = 200, batch\_size = 1, verbose = 2)

**Modelo RNN02 – LSTM Time Steps – IA**

model.add(LSTM(4, input\_shape = (None, 1)))

model.add(Dense(1))

model.compile(loss = 'mean\_squared\_error', optimizer = 'adam')

model.fit(trainX, trainY, epochs = 200, batch\_size = 1, verbose = 2)

**Modelo RNN02 – LSTM Stateful – IA**

model.add(LSTM(4, batch\_input\_shape = (batch\_size, look\_back, 1), stateful = True))

model.add(Dense(1))

model.compile(loss = 'mean\_squared\_error', optimizer = 'adam')

fori in range(200):

model.fit(trainX, trainY, epochs = 1, batch\_size = batch\_size, verbose = 2, shuffle = **False)**

model.reset\_states()

**Modelo RNN02 – LSTM Stacked – IA**

model.add(LSTM(4, batch\_input\_shape = (batch\_size, look\_back, 1), stateful = True, return\_sequences = True))

model.add(LSTM(4, batch\_input\_shape = (batch\_size, look\_back, 1), stateful = True))

model.add(Dense(1))

model.compile(loss = 'mean\_squared\_error', optimizer = 'adam')

fori in range(200):

model.fit(trainX, trainY, epochs = 1, batch\_size = batch\_size, verbose = 2, shuffle = False)

model.reset\_states()

**Modelo RNN02 – LSTM Stacked – IA (número maior de camadas)**

model = Sequential()

model.add(LSTM(100, batch\_input\_shape = (batch\_size, look\_back, 1), activation = 'relu', return\_sequences = True))

model.add(LSTM(50, activation = 'relu', return\_sequences = True))

model.add(LSTM(20, activation = 'relu'))

model.add(Dense(10, activation = 'relu'))

model.add(Dense(5, activation = 'relu'))

model.add(Dense(1))

model.compile(loss = 'mean\_squared\_error', optimizer = 'adam')

for i in range(400):

model.fit(trainX, trainY, epochs = 1, batch\_size = batch\_size, verbose = 1, shuffle = False)

model.reset\_states()

Os modelos foram baseados em cursos da Data Science Academy DSA e na timeline da Comunidade no portal:  [www.datascienceacademy.com.br](http://www.datascienceacademy.com.br/)