**Forecasting - Previsões em Séries Temporais**

**Óbitos em Internações Hospitalares - AIH**

Guttenberg Ferreira Passos

Ilan Chamovitz

Claudio Chiba

Melissa Ravanini Magalhaes

Modelagens lineares e não lineares com séries temporais.

* Para modelos não lineares usaremos Métodos Estatísticos, Inteligência Artificial, Machine Learning e Deep Learning, 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.

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

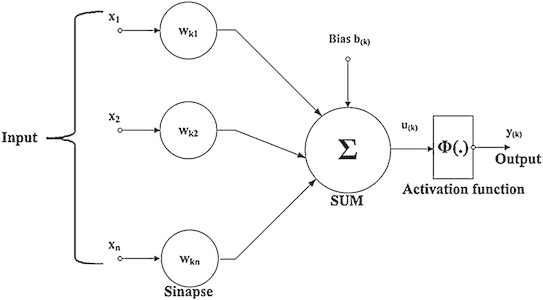
Neurônio Artificial

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.



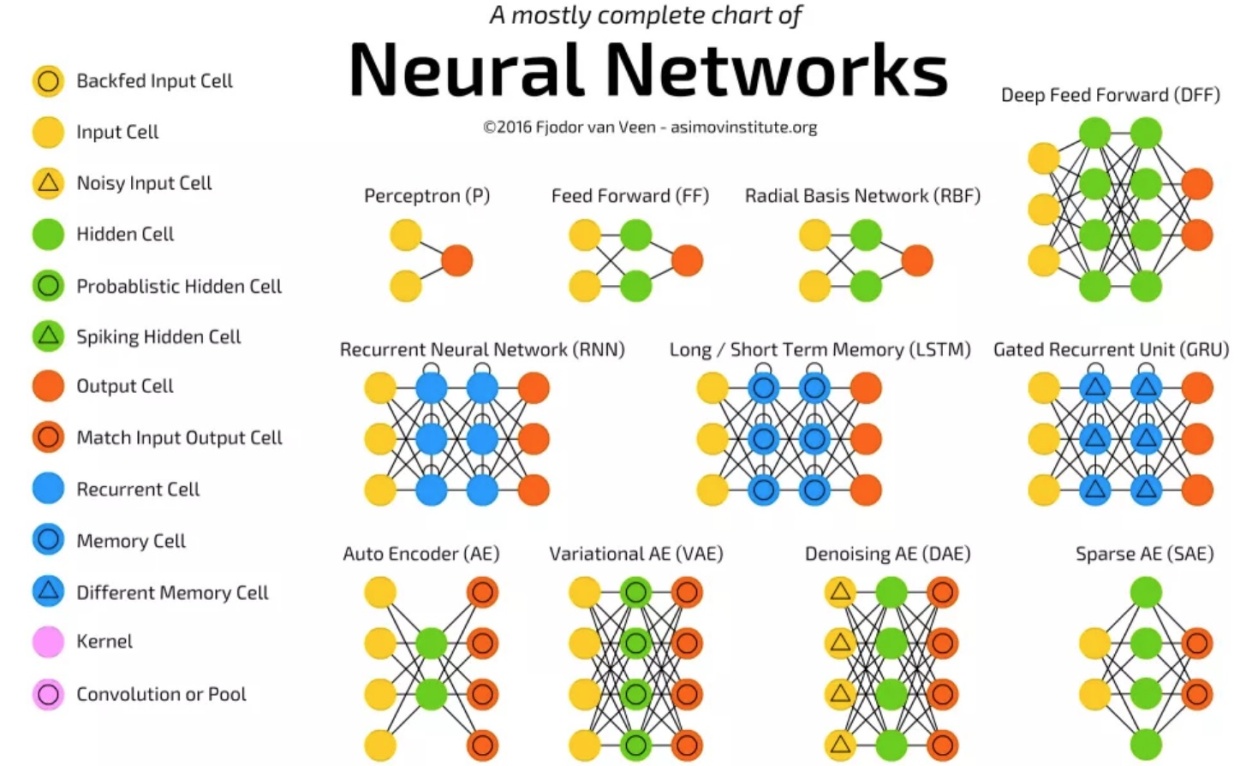
Fonte: Data Science Academy - DSA

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

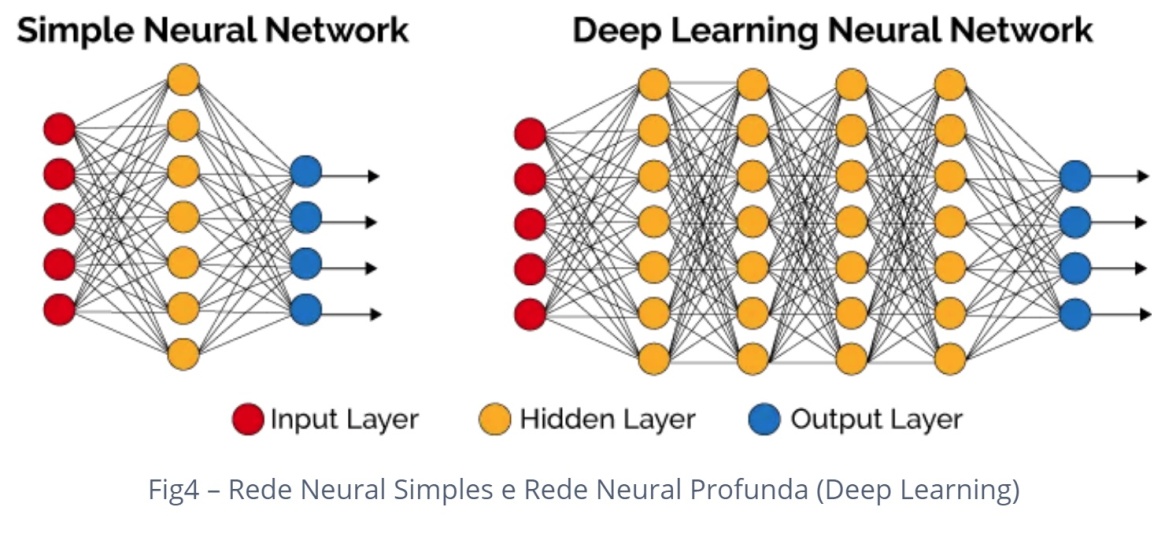
Redes neurais são sistemas de computação com nós interconectados que funcionam como os neurônios do cérebro humano. Usando algoritmos, elas podem reconhecer padrões escondidos e correlações em dados brutos, agrupá-los e classificá-los, e com o tempo aprender e melhorar continuamente.

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

Deeplearning 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 número de óbitos de pacientes acima de 50 anos nas internações hospitalares - AIH.

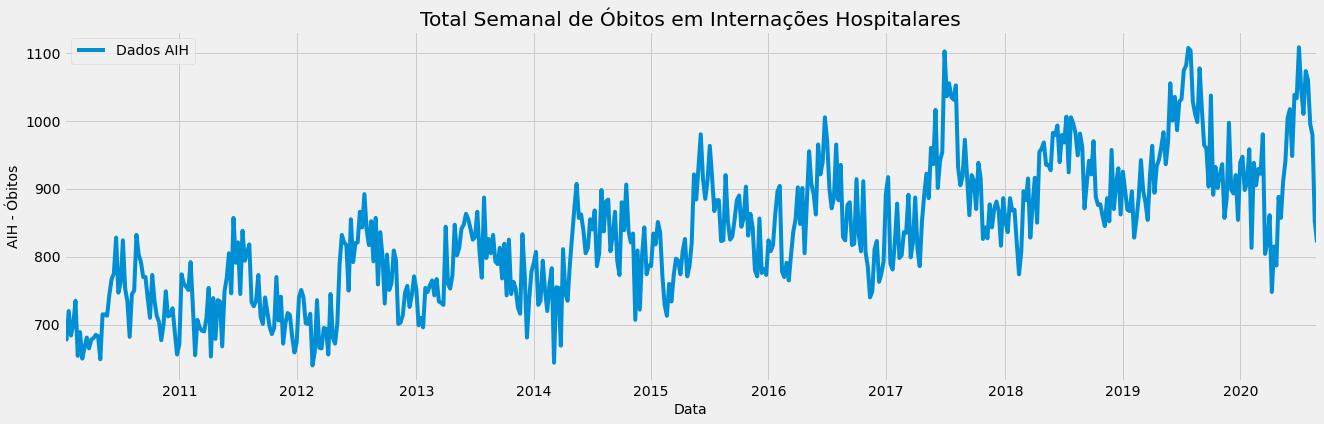
**Conjunto de Dados**

Usaremos conjuntos de dados que mostram o número de óbitos de pacientes acima de 50 anos nas internações hospitalares – AIH em Minas Gerais.

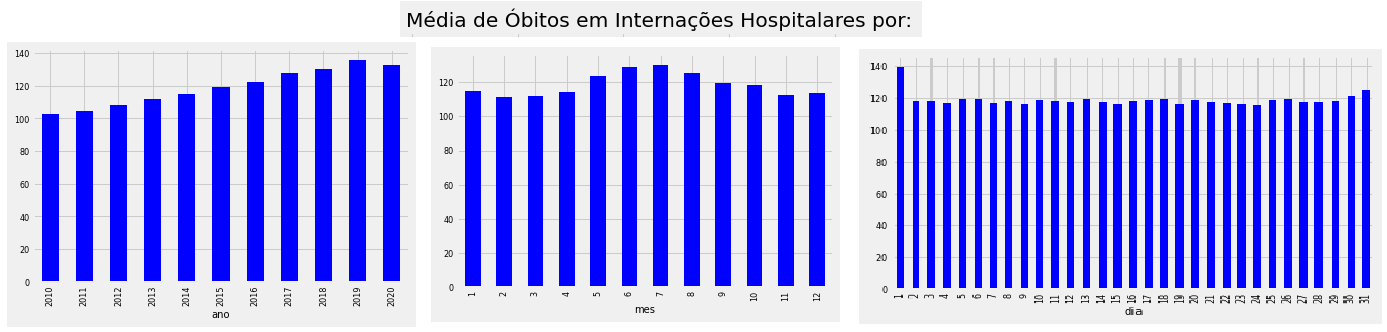
Os dados foram baixados do site do Datasus, módulo SIHSUS, tipo de arquivo RD- AIH Reduzida, anos de 2010 a 2020, UF MG, todos os meses:

<http://www2.datasus.gov.br/DATASUS/index.php?area=0901&item=1&acao=25>

Foi realizado o download de 129 arquivos (vide quadro: quantidade de Internações de pessoas acima de 50 anos que vieram a óbito), com 5.231.829 de registros utilizados para preparar um arquivo csv com o agrupamento das internações das pessoas acima de 50 anos que vieram a óbito, perfazendo um total de 3.896 registros diários, 556 registros por semana (excluindo-se a primeira e a última), com data, número e valor total das internações, dados desde 2010.

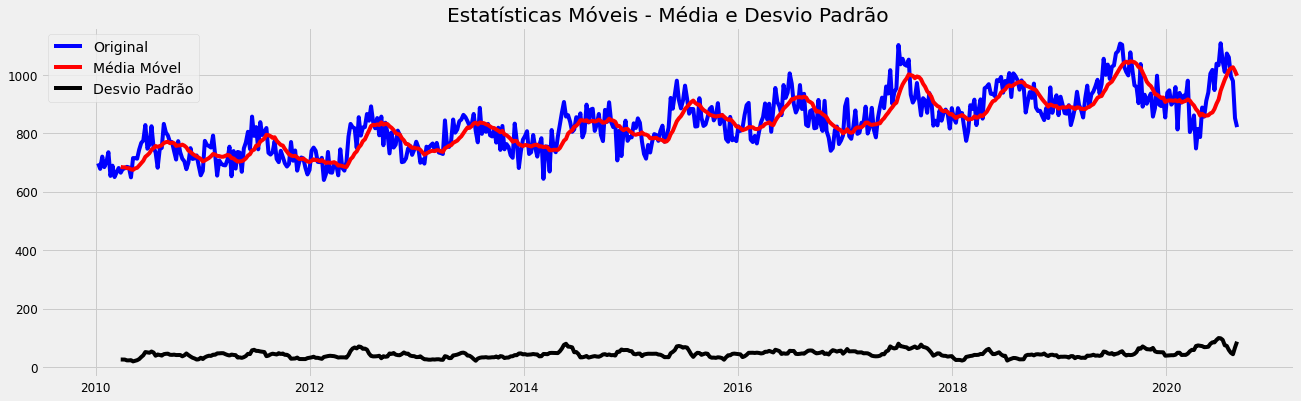


Percebemos que há uma tendência de aumento o número de óbitos de pacientes acima de 50 anos ao longo do tempo, entretanto há uma redução em 2020.

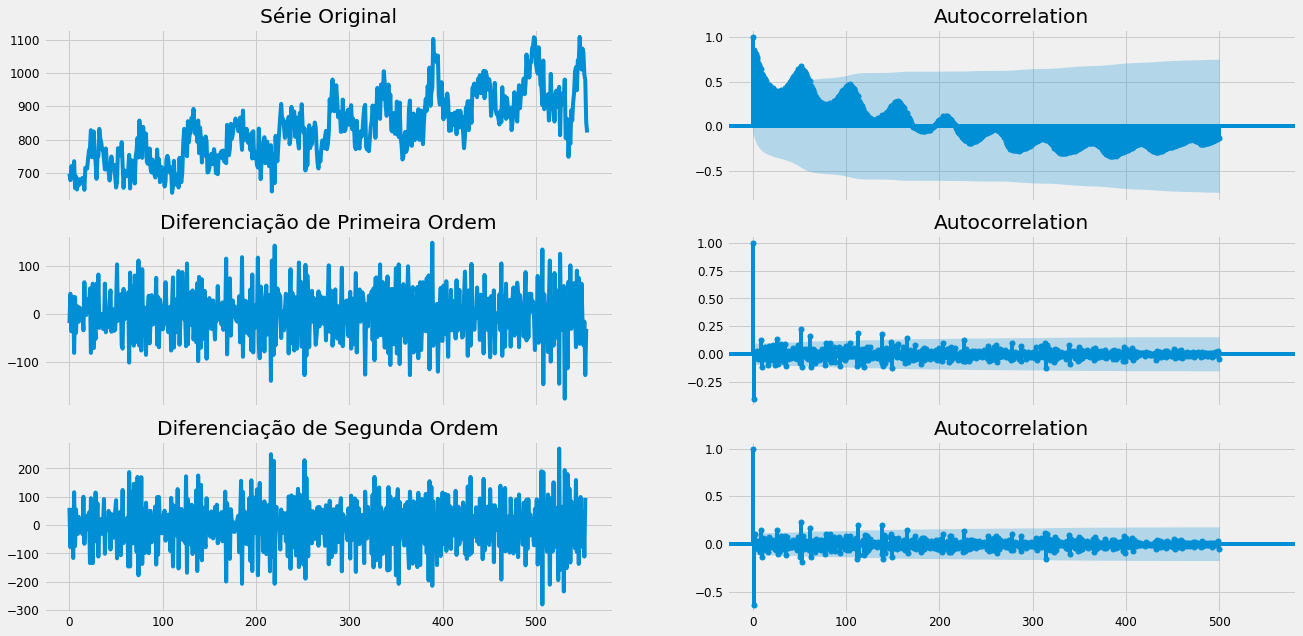


**Análise Exploratória dos Dados**

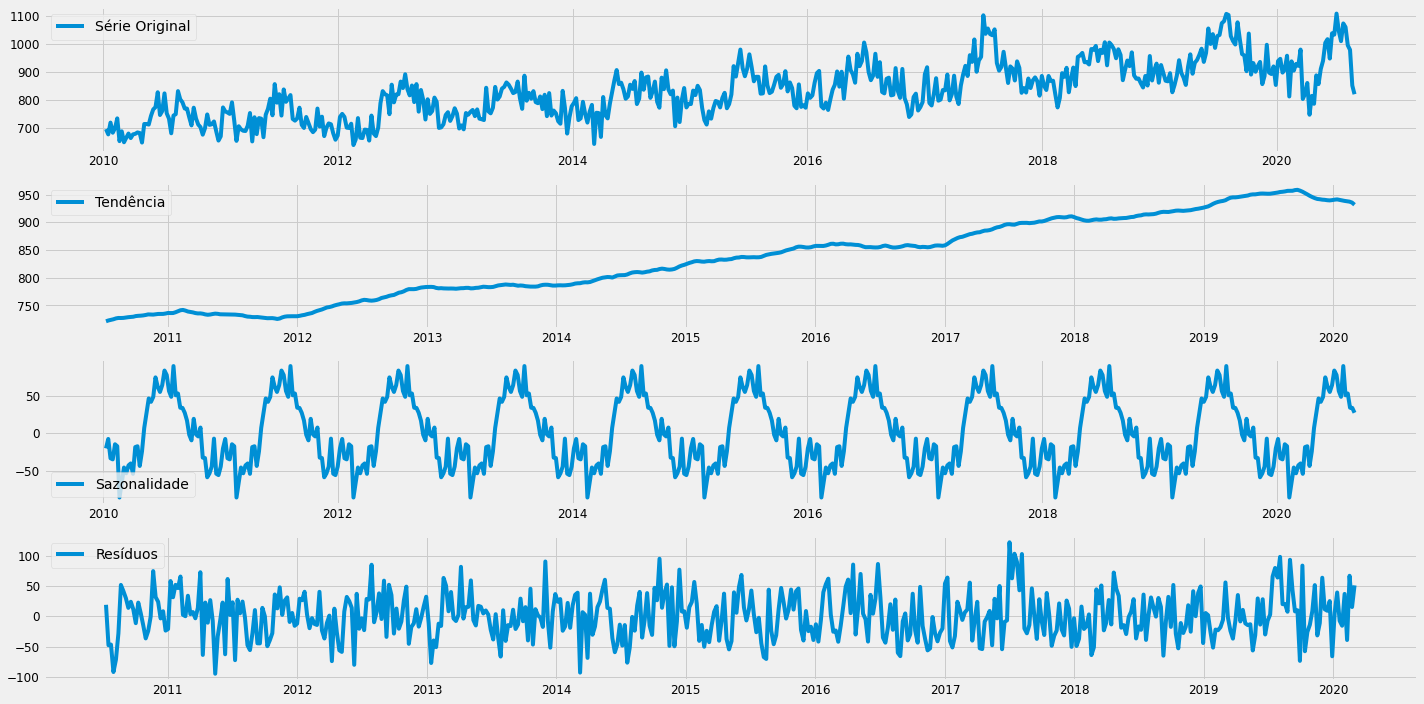
Vamos testar a estacionaridade da série.



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



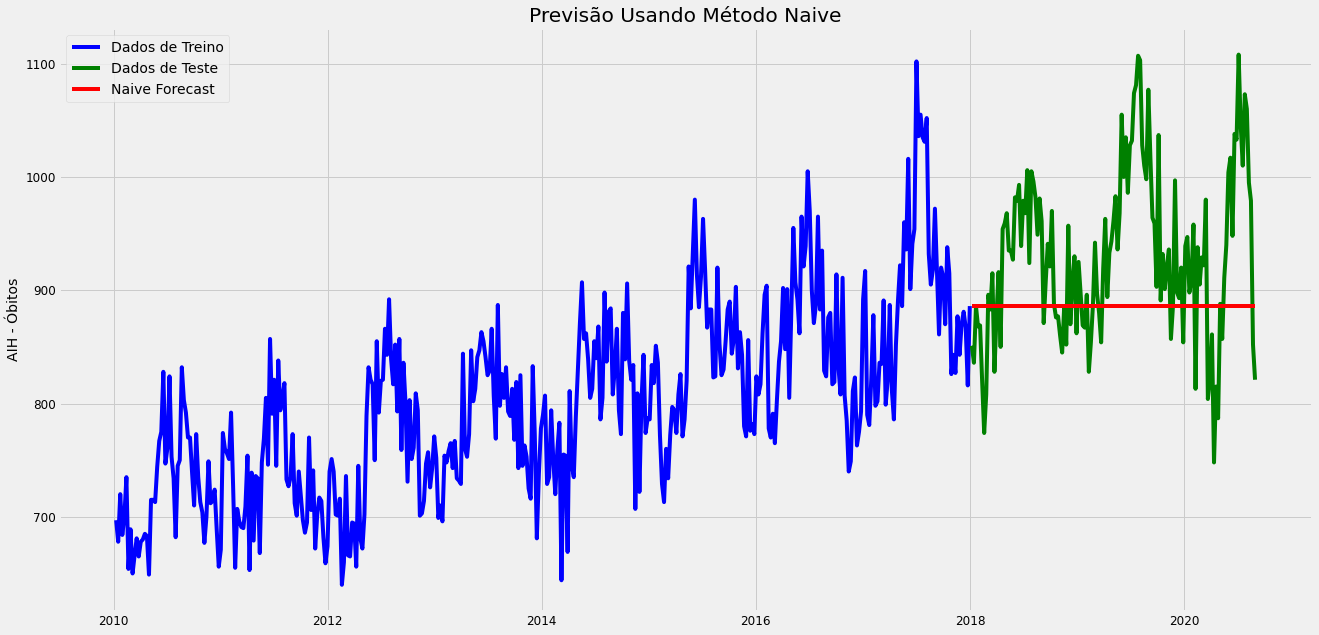
O gráfico abaixo permite a visualização da série original, tendência, sazonalidade e resíduos:



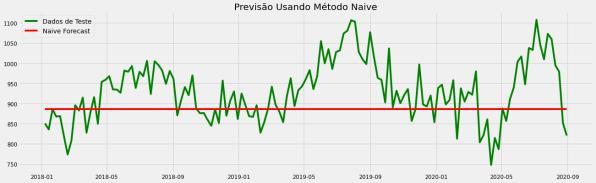
Para a execução dos modelos utilizando **Métodos Estatísticos** os dados foram separados em:

* 417registros de treino e
* 139 registros de validação.

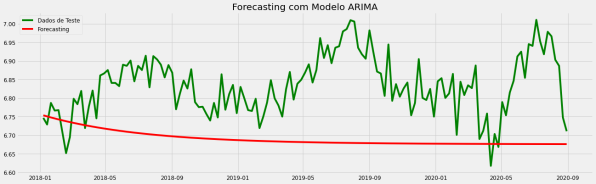
**Modelo 11 – Previsões Método Naive**

****

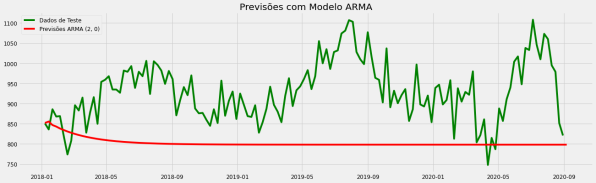
**11 – Método Naive**



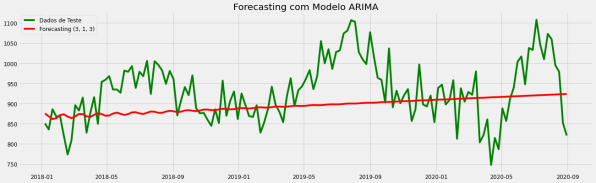
**13 - Forecasting–ARIMALOG (1, 0, 1)**



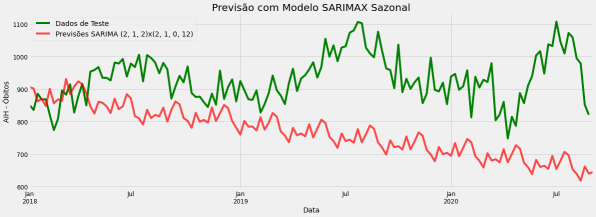
**15 –ARMA (2, 0)**

****

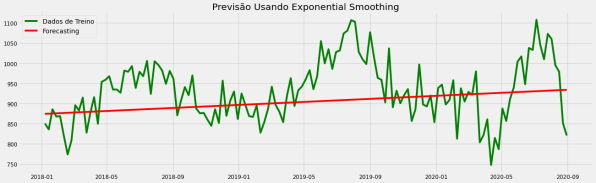
**15 – Forecasting - ARIMA (3, 1, 3)**

****

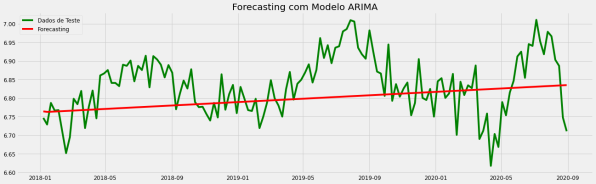
**17 – SARIMAX(2,1,1)(2,1,0,12)**

****

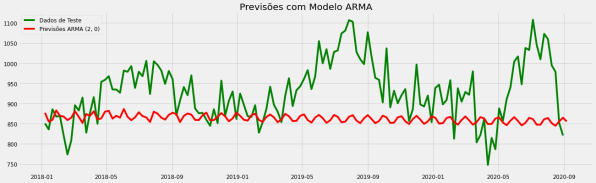
**12 – ExponentialSmoothing**



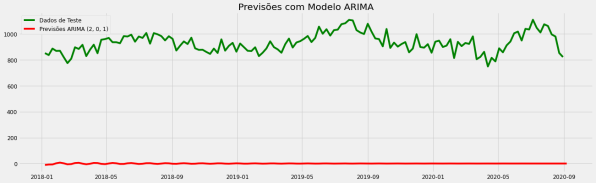
**14 - Forecasting– ARIMALOG (1, 1, 1)**



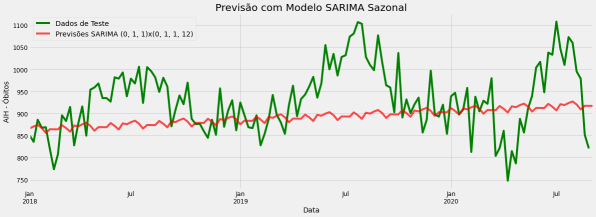
**15 – ARMA (12, 9)**



**16 - Predict-ARIMA (3, 1, 3)**



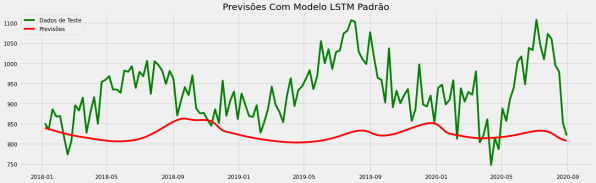
**18 – SARIMAX(0,1,1)(0,1,1,12)**

****

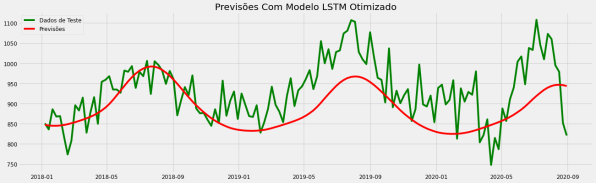
**Modelos utilizando**

**Inteligência Artificial - IA**

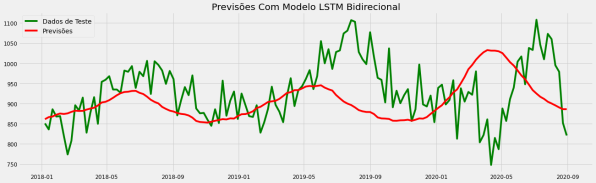
**22 – LSTM - IA (3 repetições)**

****

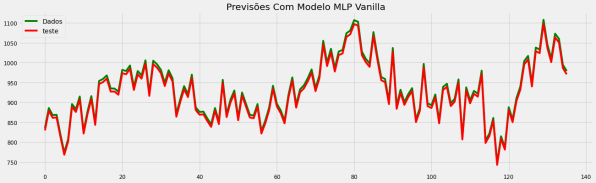
**23 – LSTM otimizado - IA**

****

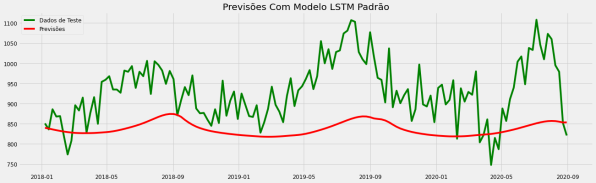
**25 – LSTM Bidirecional– IA**

****

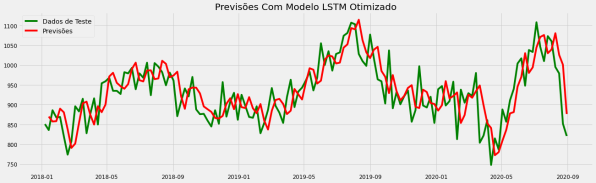
**RNN01 – MLP Vanilla – IA –**

****

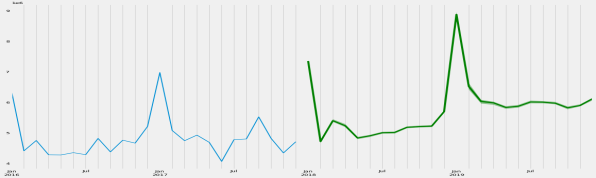
**22 – LSTM - IA (5 repetições)**

****

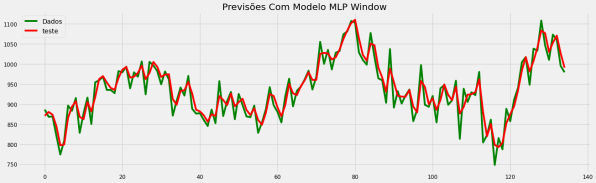
**24 – Stacked LSTM - IA**

****

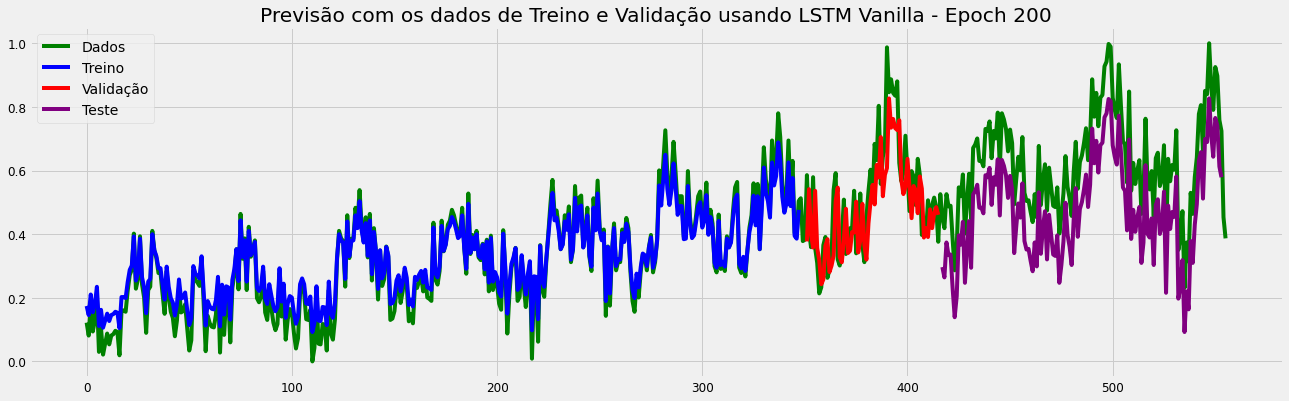
**25 – DeepAR – IA**

****

**RNN01 – MLP Window – IA –**

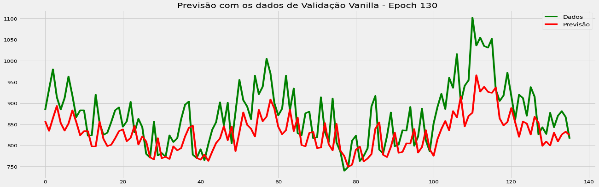
****

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

****

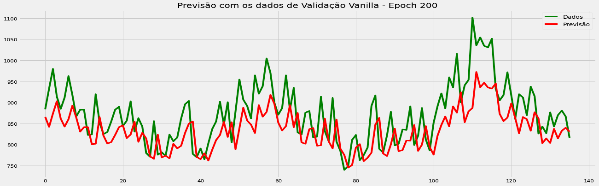
**RNN02 – LSTM Vanilla IA – Validação**

**Epoch = 130 - evita o Overfitting**

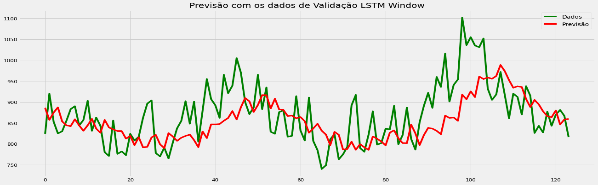
****

**RNN02 – LSTM Vanilla IA – Validação**

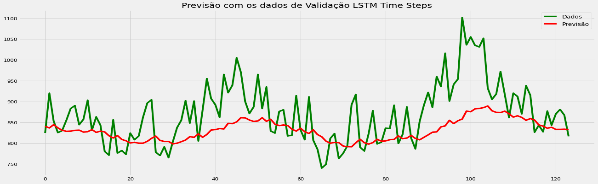
**Epoch = 200**

****

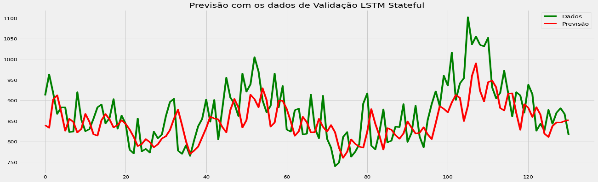
**RNN02 – LSTM Window IA– Validação**

****

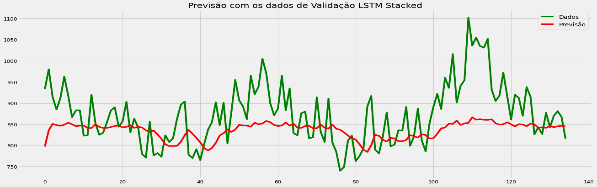
**RNN02– LSTM Time Steps IA Validação**

****

**RNN02 – LSTM Stateful IA – Validação**

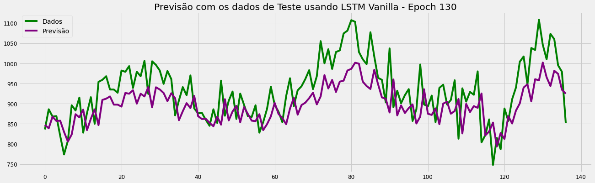
****

**RNN02 – LSTM Stacked IA – Validação**

****

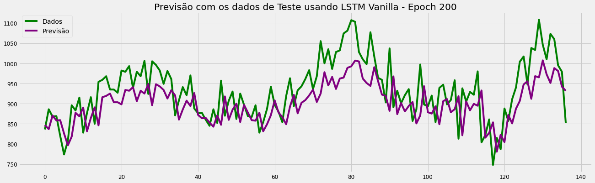
**RNN02 – LSTM Vanilla IA – Teste**

**Epoch = 130 - evita o Overfitting**

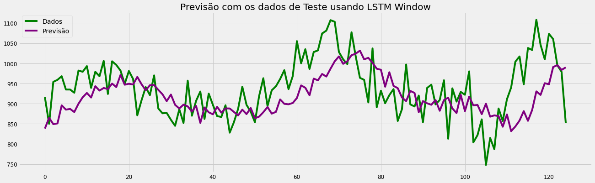
****

**RNN02 – LSTM Vanilla IA – Teste**

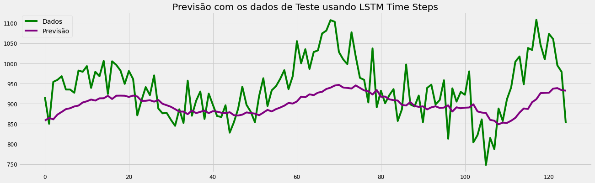
**Epoch = 200**

****

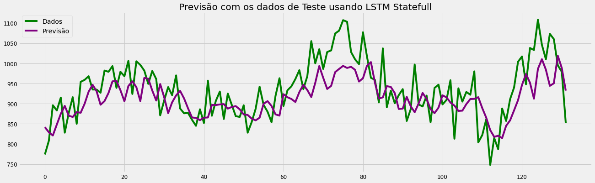
**RNN02 – LSTM Window IA – Teste**

****

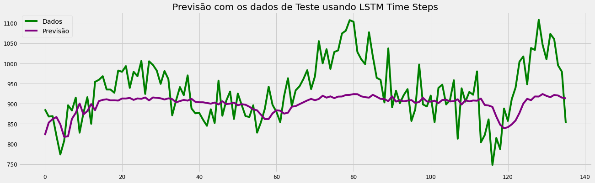
**RNN02 – LSTM TimeSteps IA – Teste**

****

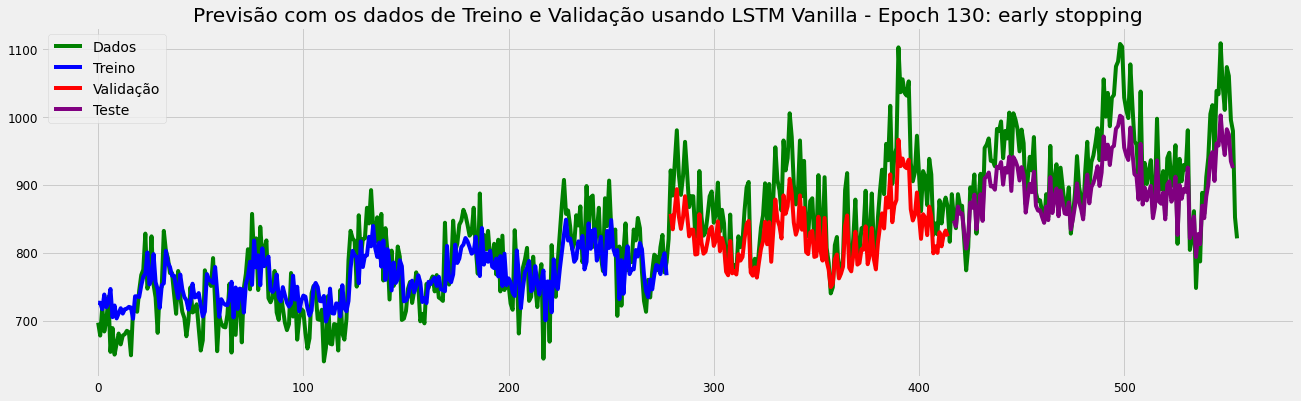
**RNN02 – LSTM Stateful IA – Teste**

****

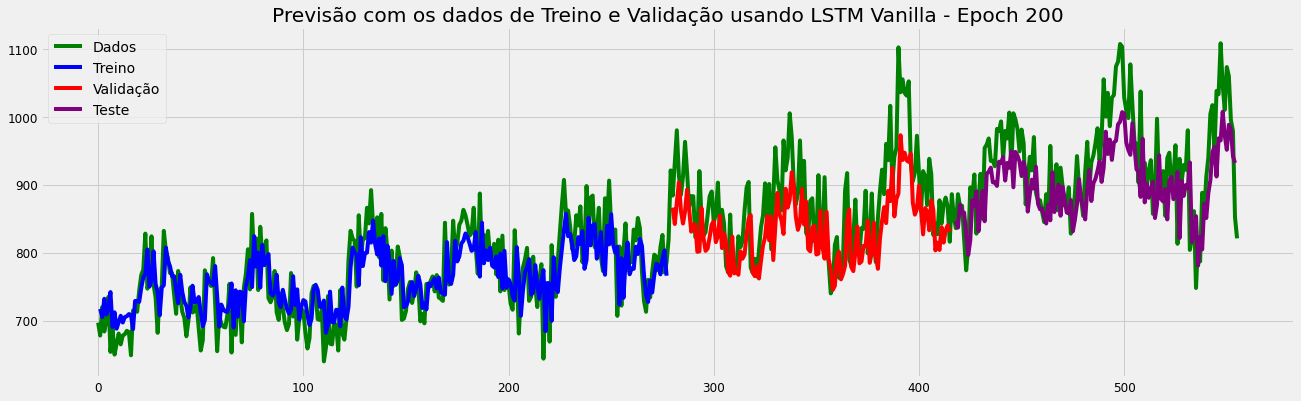
**RNN02 – LSTM Stacked IA – Teste**

****

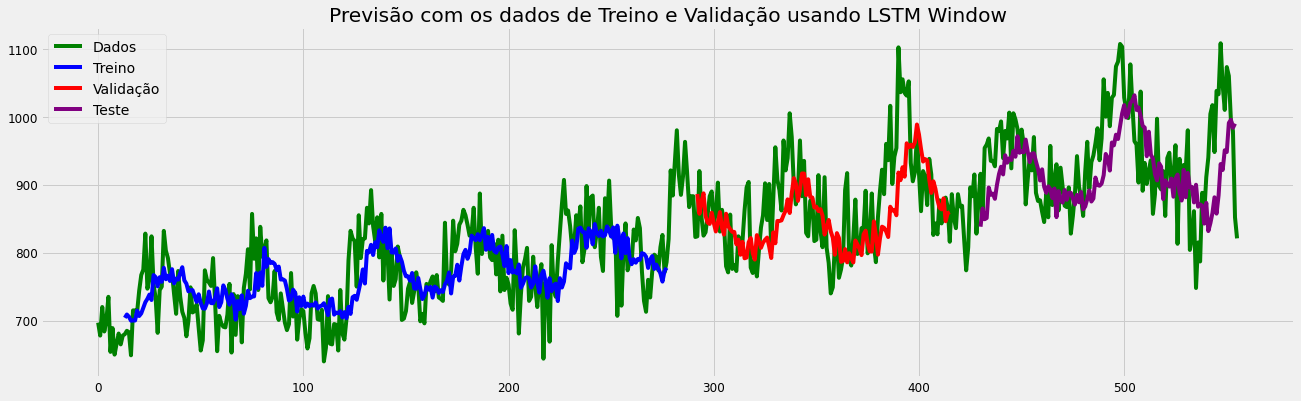
**Modelo RNN02 – LSTM Vanilla IA – Epoch = 130 – evita o Overfitting**

****

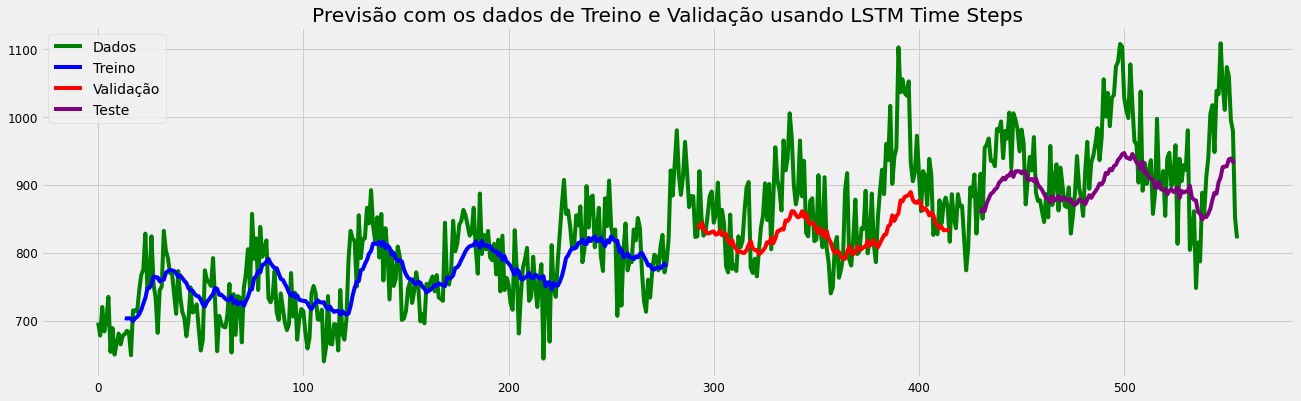
**Modelo RNN02 – LSTM Vanilla IA – Epoch = 200**

****

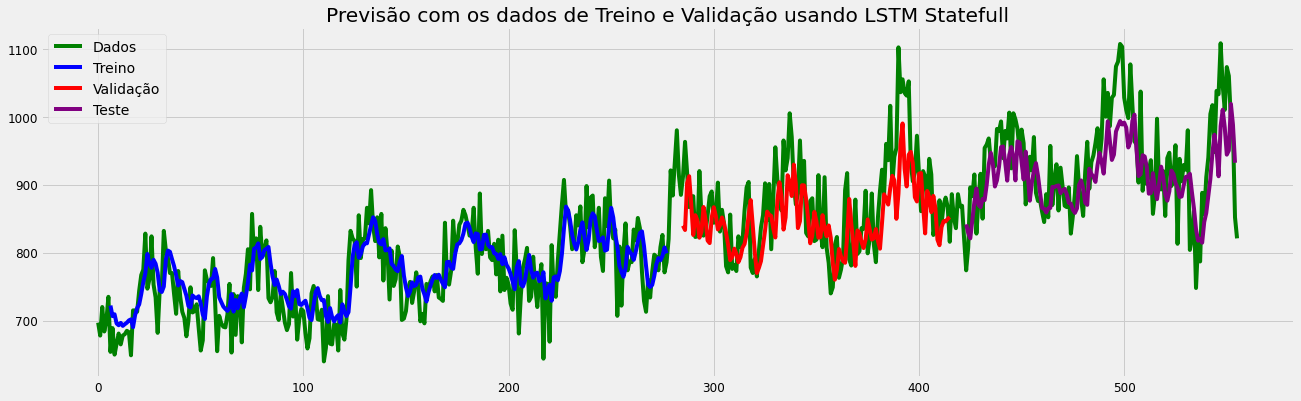
**ModeloRNN02 – LSTM Window IA**



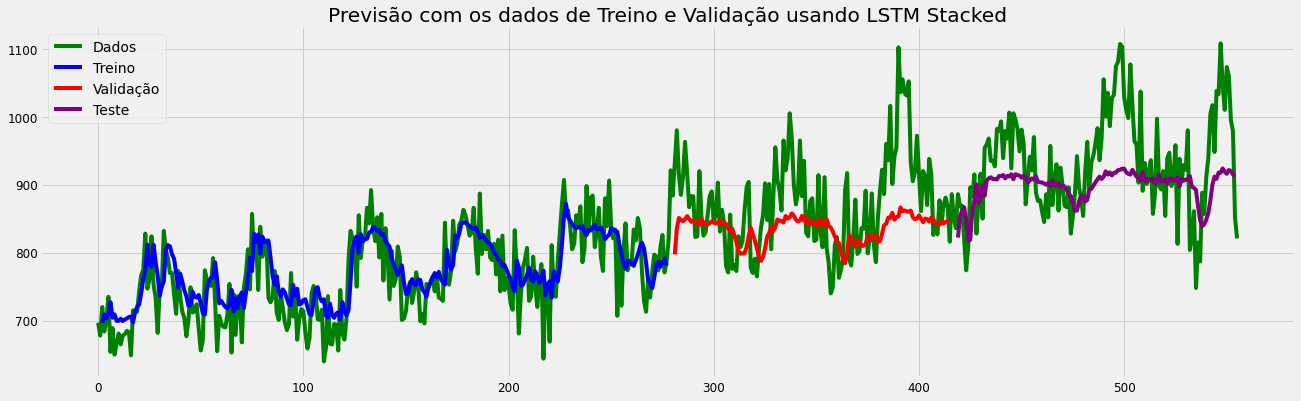
**ModeloRNN02– LSTM Time Steps**

****

**ModeloRNN02– LSTM Stateful**

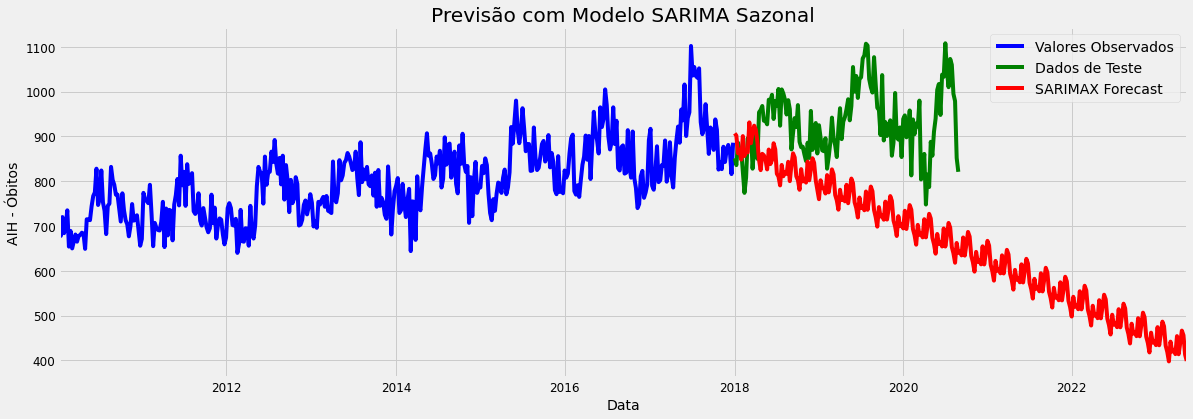
****

**Modelo RNN02– LSTM Stacked**

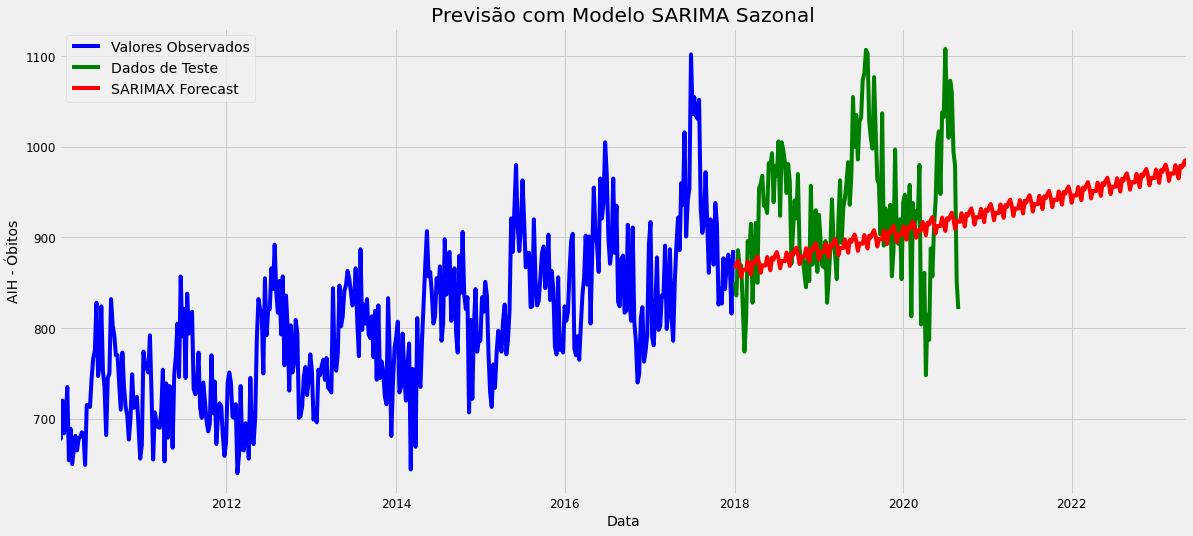
****

**Forecasting utilizando Inteligência Artificial - IA**

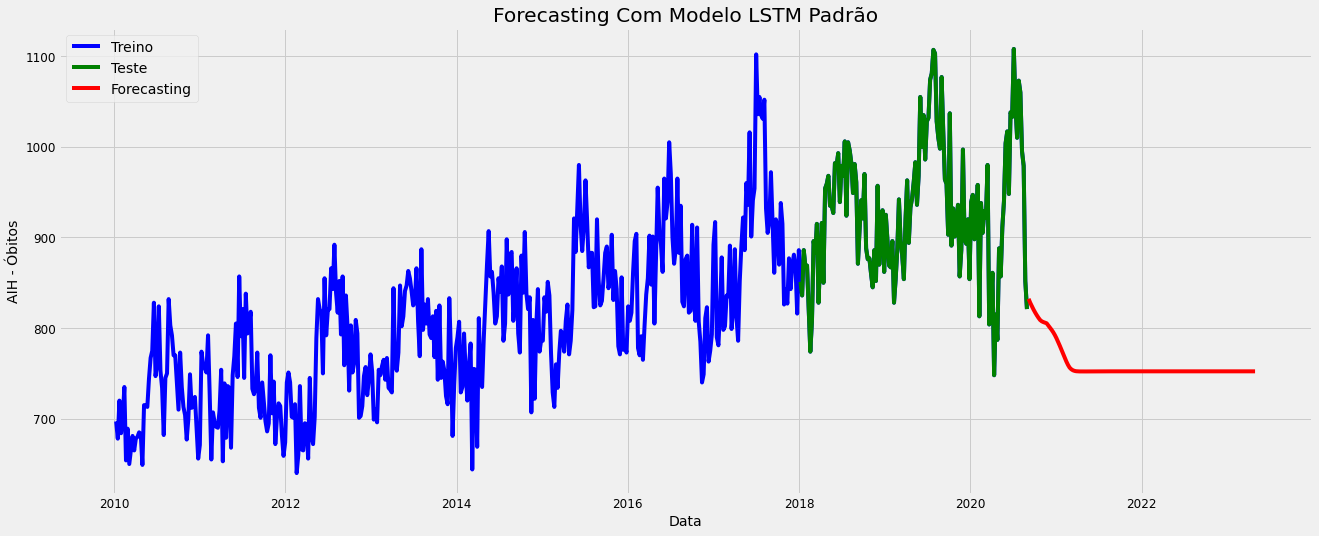
**Modelo 17 –SARIMA (2, 1, 2)x(2, 1, 0, 12)**

****

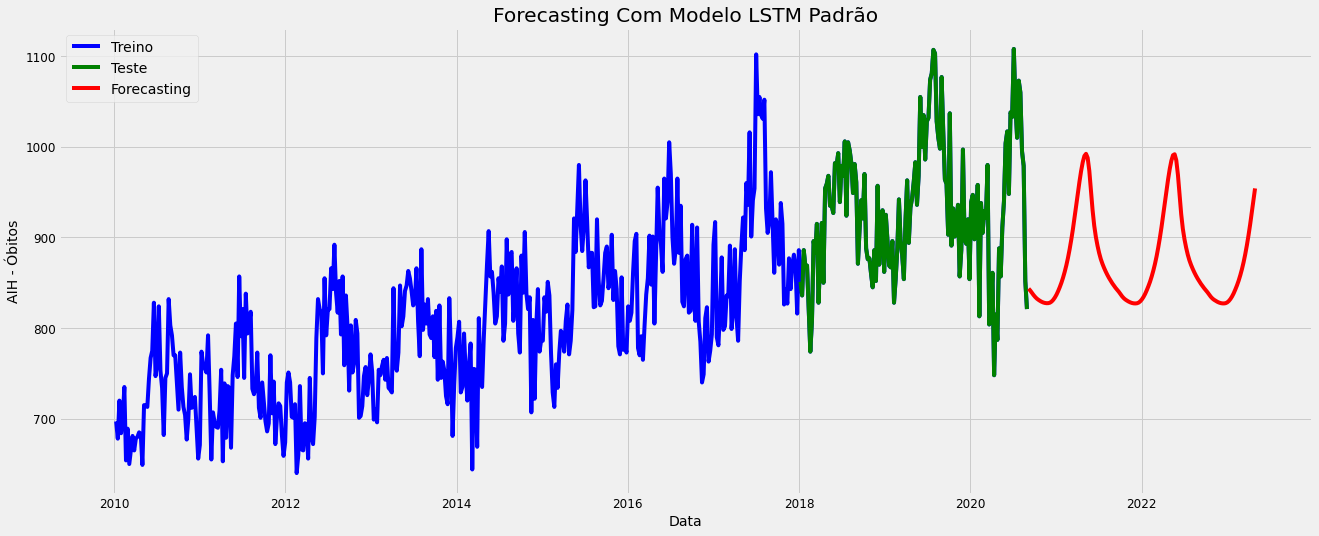
**Modelo 18 –SARIMA (0, 1, 1)x(0, 1, 1, 12)**

****

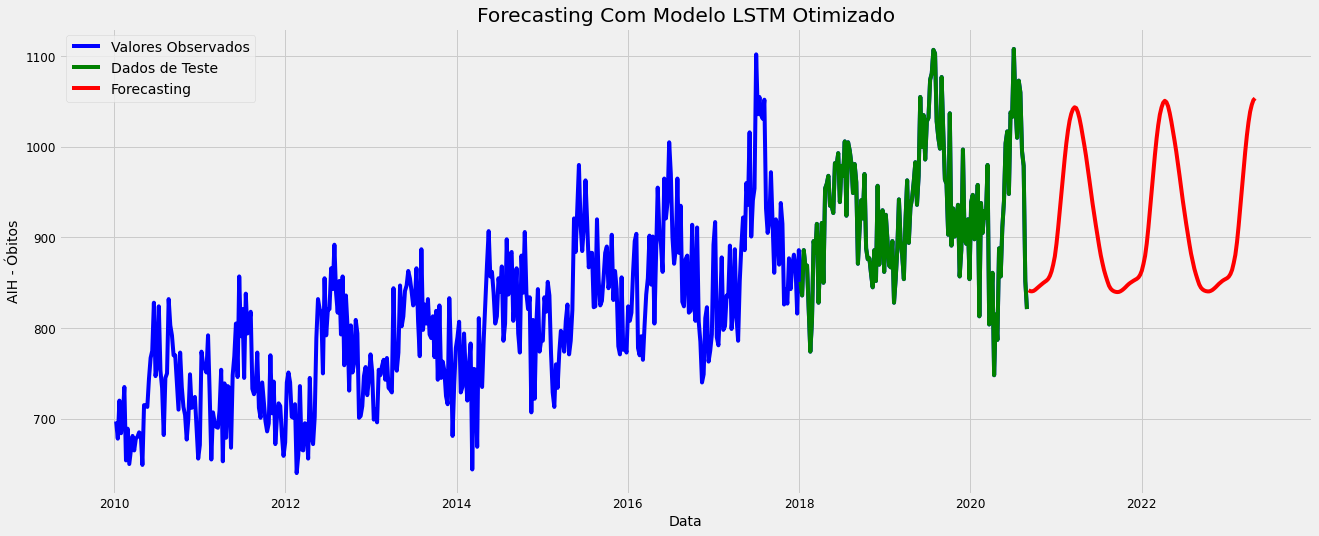
**Modelo 22 – LSTM - IA (3 repetições)**

****

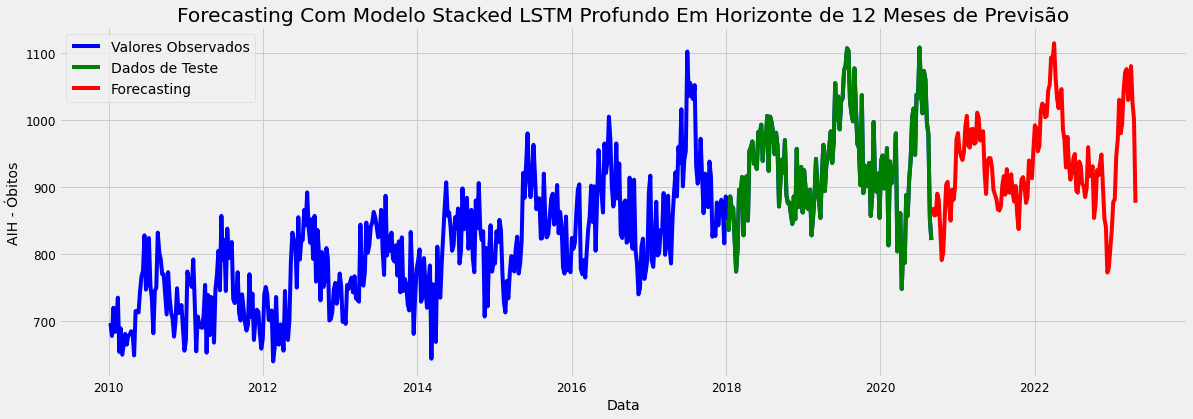
**Modelo 22 – LSTM - IA (5 repetições)**

****

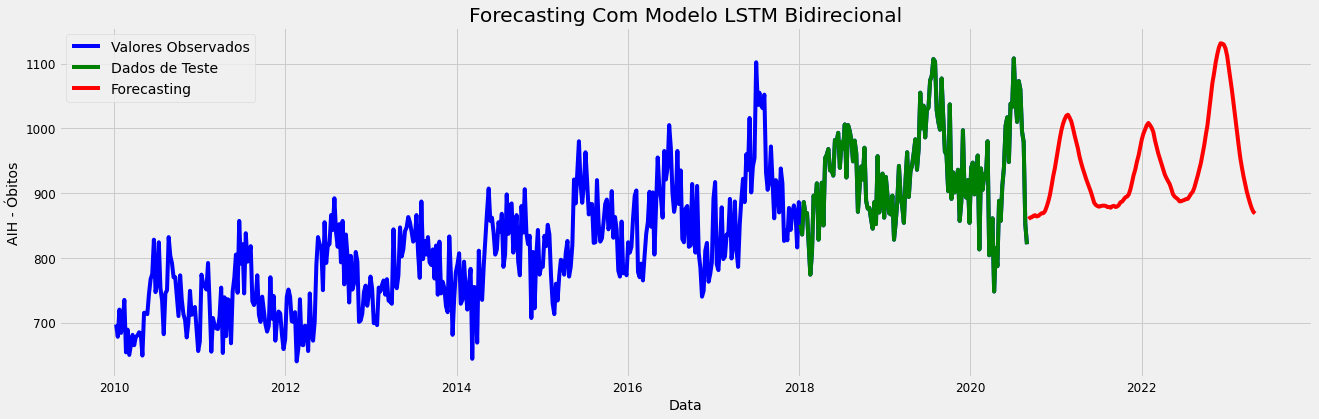
**Modelo 23 – LSTM otimizado - IA**

****

**Modelo 24 – Stacked LSTM - IA**

****

**Modelo 25 – LSTM Bidirecional – IA**

****

**Resultados dos Modelos:**







**Arquiteturas dos Modelos utilizando IA**

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

Número de repetições = 20

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)

**Modelo 25 – LSTM Bidirecional - IA**

Número de repetições = 20

modelo\_lstm.add(Bidirectional(LSTM(41, activation = 'relu'), input\_shape = (41, 1)))

modelo\_lstm.add(Dense(1))

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

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

lista\_hiperparametros():

n\_input = [24]

n\_nodes = [100]

n\_epochs = [200]

n\_batch = [5]

n\_diff = [12]

Número de repetições = 20

modelo\_lstm.add(Bidirectional(LSTM(100, activation = 'relu'), input\_shape = (41, 1)))

modelo\_lstm.add(Dense(1))

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

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

**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()

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/)