Forecasting - Previsões em Séries Temporais

Arrecadação de ICMS

Guttenberg Ferreira Passos

Modelagens lineares e não lineares com séries temporais.

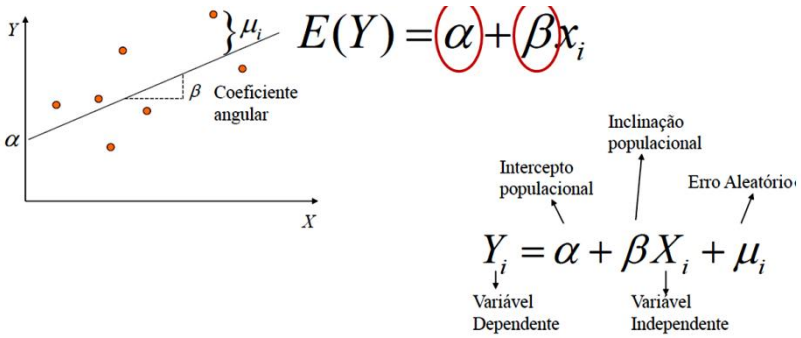
- Para modelos lineares usaremos Estatística com Análise de Tendências;
- Para modelos não lineares usaremos Métodos Estatísticos, Inteligência Artificial, Machine Learning e Deep Learning, algoritmos:
  - Naive;
  - Exponential Smoothing;
  - ARMA - Autoregressive Moving Average;
  - ARIMA - Autoregressive Integrated Moving Average;
  - SARIMAX – Seasonal Autoregressive Integrated Moving Average;
  - DeepAR – GluonTS - Probabilistic Time Series Modeling;
  - MLP - Multilayer Perceptron;
  - LSTM - Long short-term memory.

A Análise de Tendências é um aspecto da análise de negócios que tenta prever o movimento futuro de um produto ou serviço com base nos dados históricos e estatísticos. Com isso, é possível definir estratégias de atuação, planos de ação e tomada de decisão.

Uma dessas tendências é a Análise de Regressão, que estuda a relação entre uma variável dependente e outras independentes. A relação entre elas é representada por um modelo matemático. Este modelo é designado por Modelo de Regressão Linear Simples (MRLS), o qual define uma relação linear entre a variável dependente e uma variável independente.

Modelo de Regressão Linear Simples MRLS:

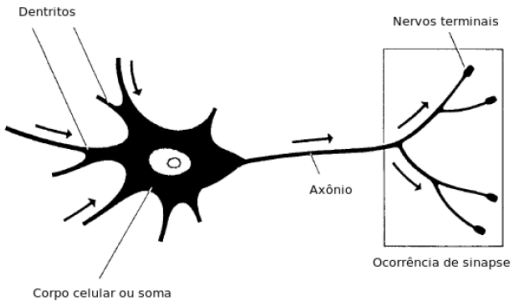
$$Y_i = \alpha + \beta X_i + \mu_i$$



Neurônio Biológico

O Deep Learning Book Brasil é uma iniciativa da Data Science Academy, 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.

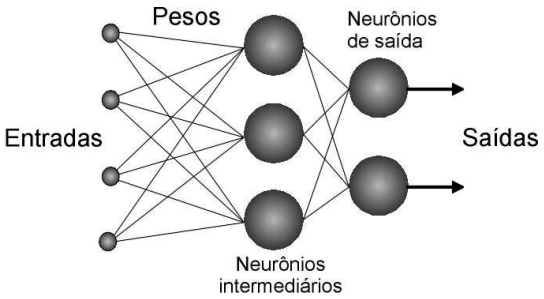
De acordo o com o Deep Learning Book, o neurônio biológico é uma célula, que pode ser dividida em três seções: o corpo da célula, os dendritos e o axônio, cada uma com funções específicas, porém complementares.



Fonte: Data Science Academy - DSA

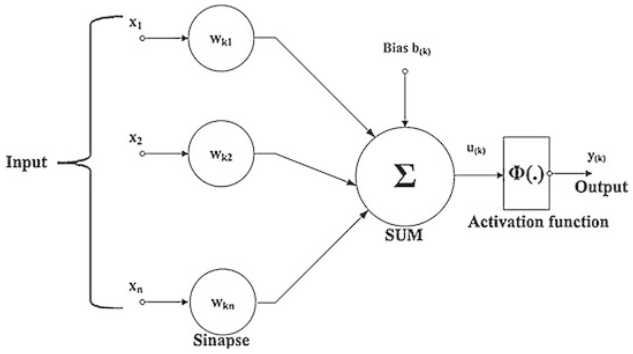
Neurônio Artificial

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.



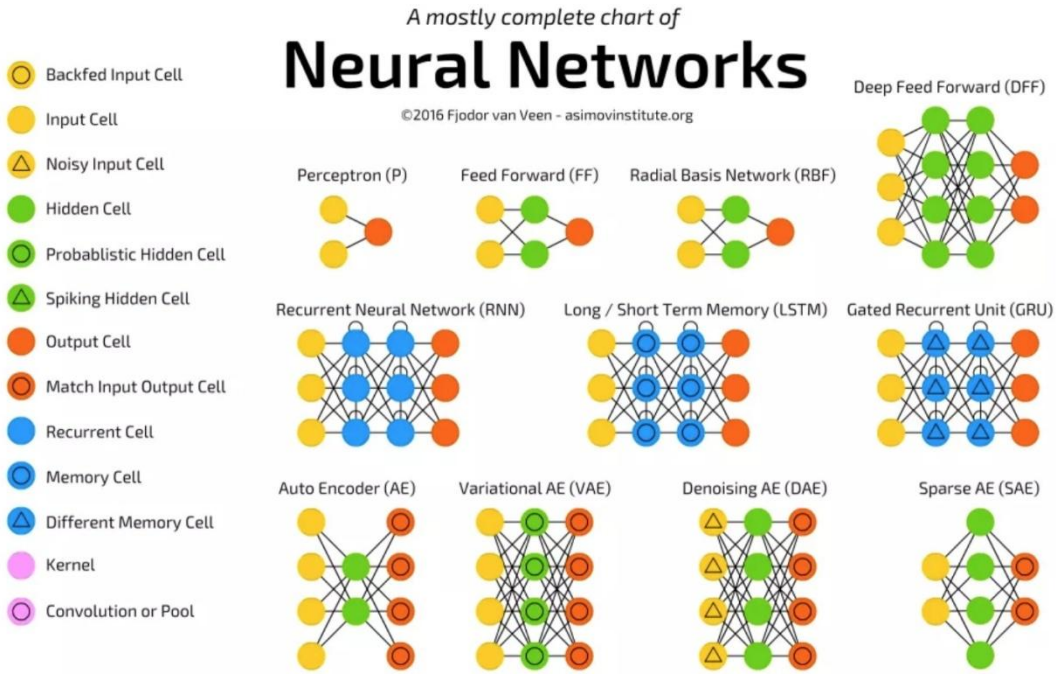
Fonte: Data Science Academy - DSA

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), Feed Foward (FF), Recurrent Neural Network (RNN) e Long Short Term Memory (LSTM):



Fonte: THE ASIMOV INSTITUTE

Deep learning é uma das bases da Inteligência Artificial (IA), um tipo de aprendizado de máquina (Machine Learning) 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:

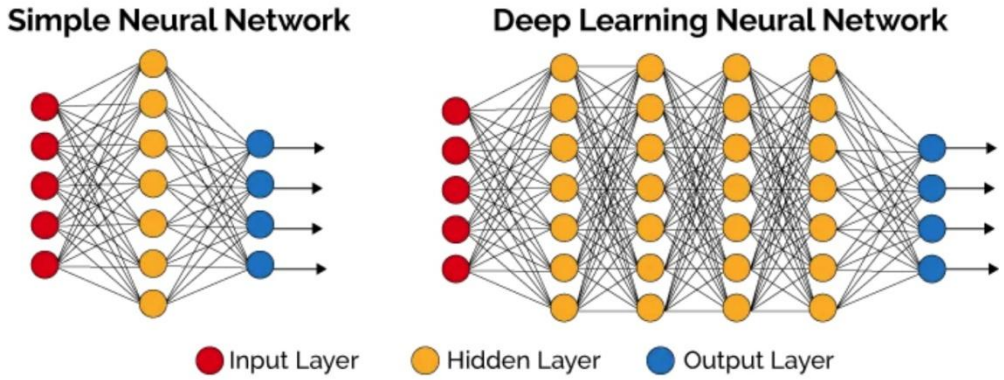


Fig4 – Rede Neural Simples e Rede Neural Profunda (Deep Learning)

Modelo Base

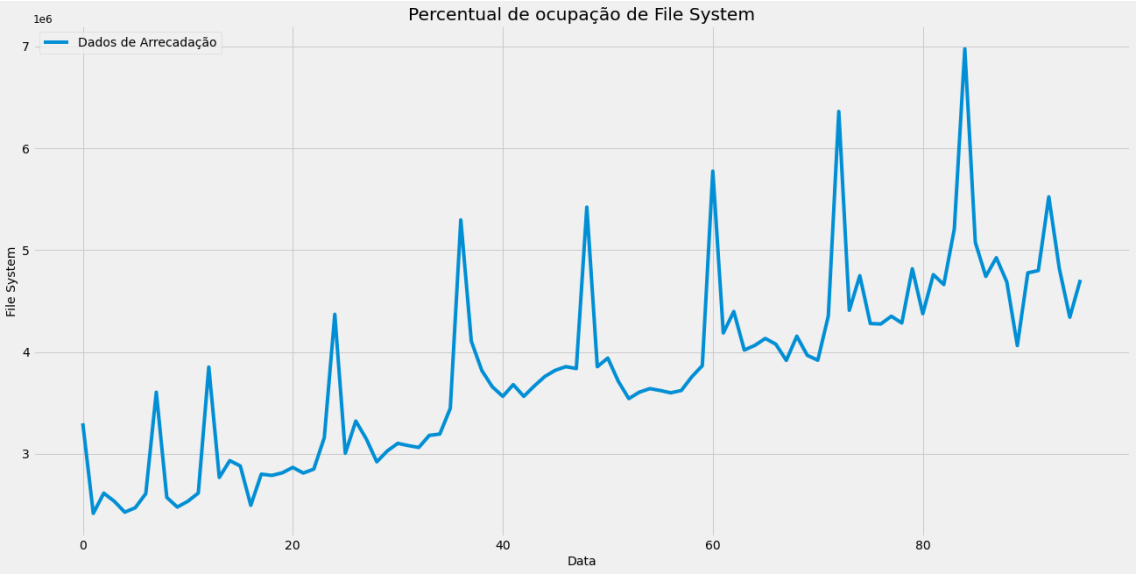
Definição do Problema de Negócio

Previsão de arrecadação de ICMS.

Conjunto de Dados

Usaremos conjuntos de dados que mostram a arrecadação de ICMS. Os dados tem registros dos anos de 2010 a 2015.

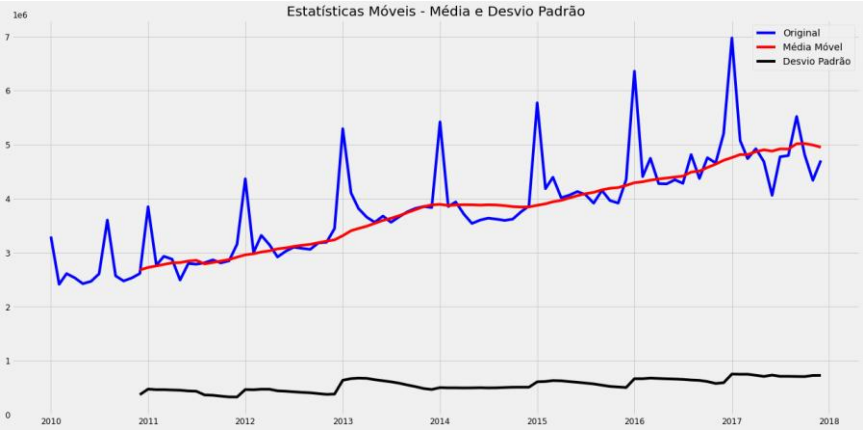
Base de dados com 1 dataset com 2 colunas, data e arrecadação de ICMS com 96 registros:



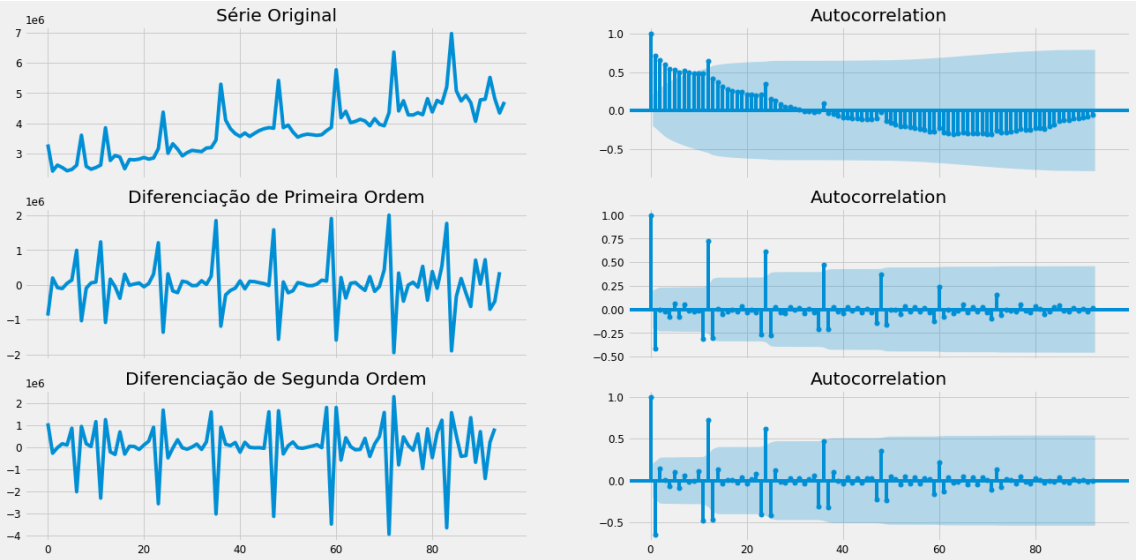
Percebemos que há uma tendência de aumento da arrecadação de ICMS ao longo do tempo.

Análise Exploratória dos Dados

Vamos testar a estacionaridade da série.



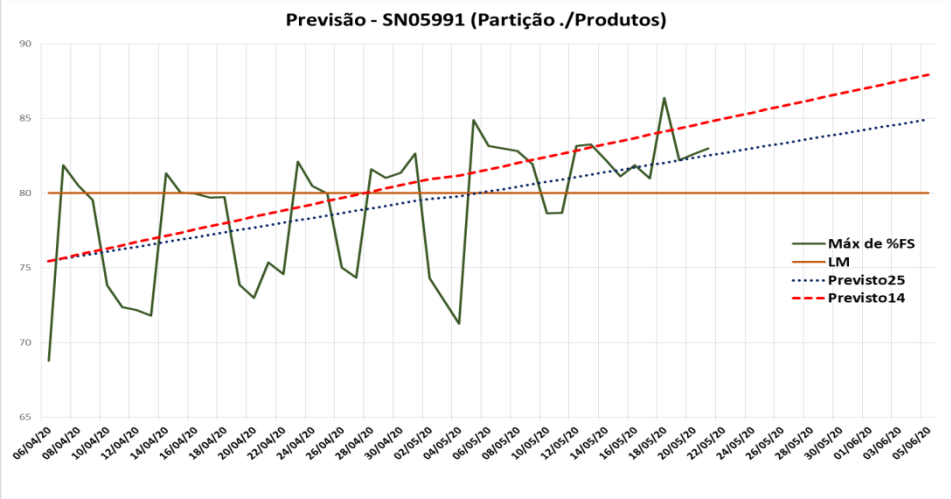
O gráfico ACF permite a avaliação da diferenciação mínima necessária para obter uma série estacionária (Parâmetro d para o Modelo ARIMA):



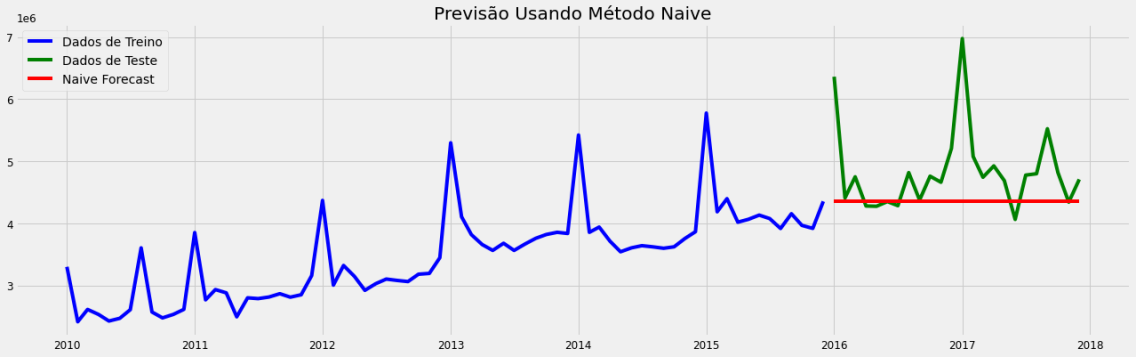
Para a execução dos modelos utilizando **Métodos Estatísticos** os dados foram separados em:

- 72 registros de treino e
- 24 registros de validação.

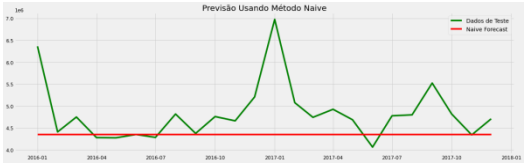
Modelo 01 – Previsões Método de Regressão Linear Simples MRLS



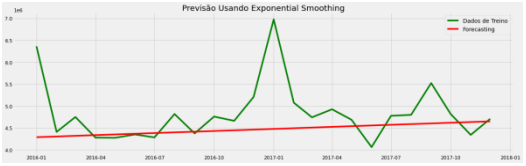
Modelo 11 – Previsões Método Naive



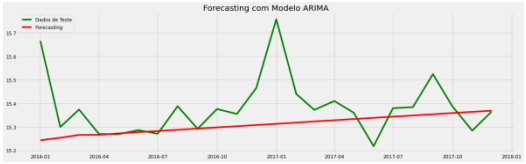
11 – Método Naive



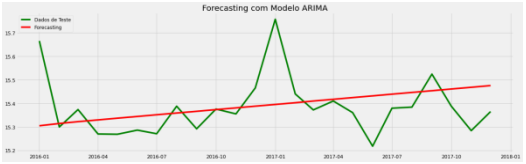
12 – Exponential Smoothing



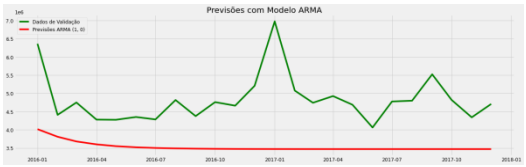
13 - Forecasting – ARIMA LOG (2, 1, 0)



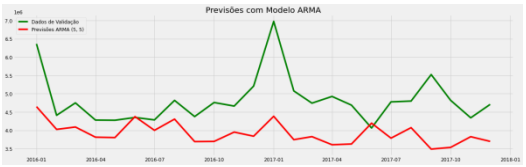
14 - Forecasting – ARIMA LOG (1, 1, 1)



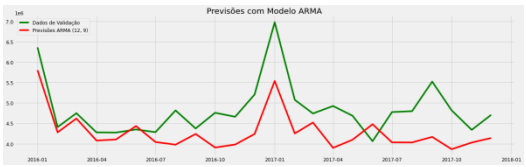
15 – ARMA (1, 0)



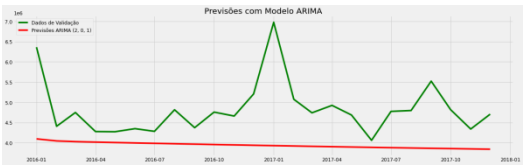
15 – ARMA (5, 5)



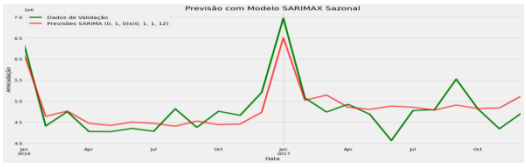
15 – ARMA (12, 9)



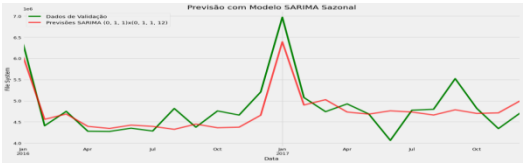
16 - Predict - ARIMA (2, 0, 1)



17 – SARIMAX (0,1,0)(0,1,1,12)



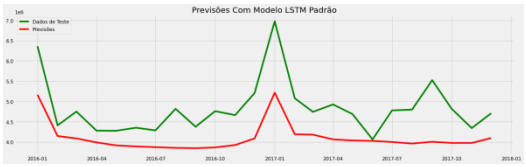
18 – SARIMAX (0,1,1)(1,1,1,12)



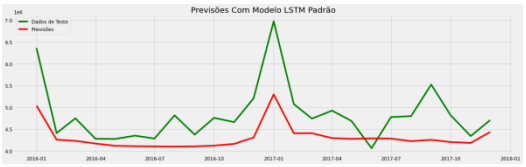
Modelos utilizando

Inteligência Artificial - IA

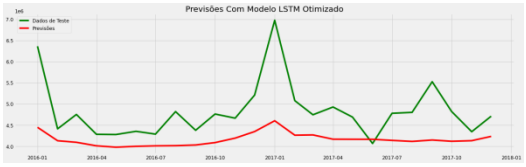
22 – LSTM - IA (5 repetições)



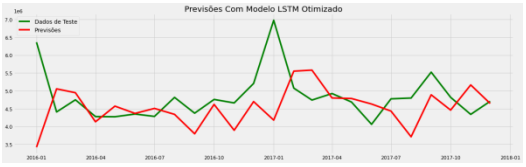
22 – LSTM - IA (20 repetições)



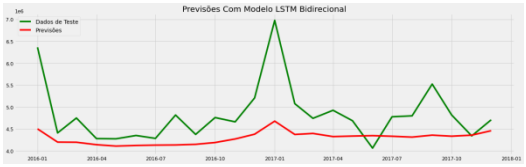
23 – LSTM otimizado - IA



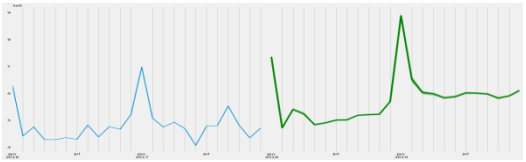
24 – Stacked LSTM - IA



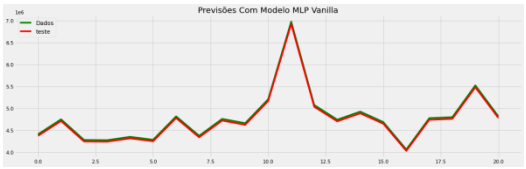
25 – LSTM Bidirecional – IA



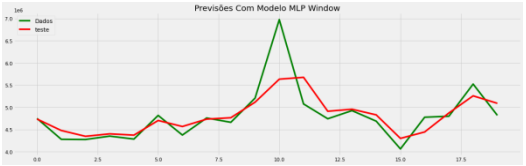
25 – DeepAR – IA



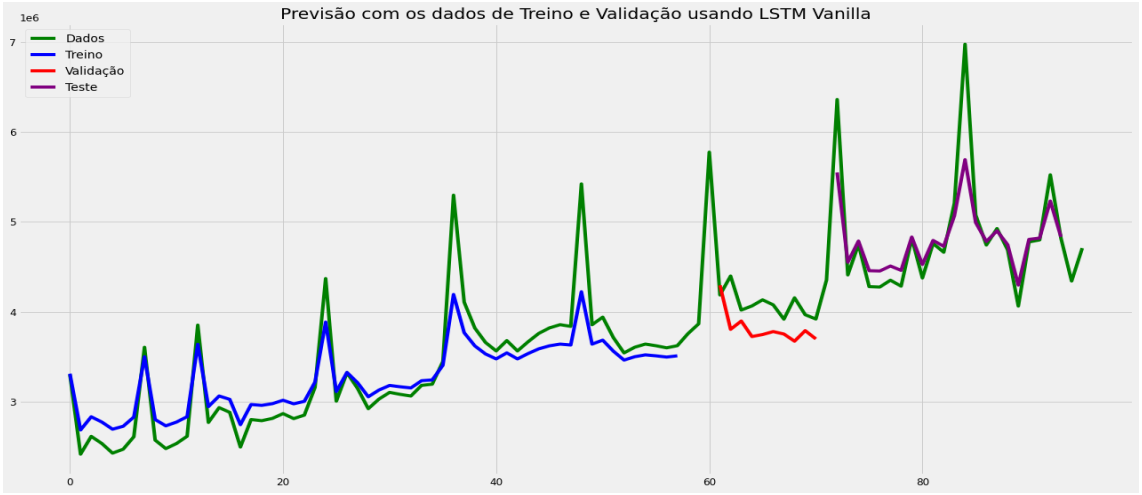
RNN01 – MLP Vanilla – IA –



RNN01 – MLP Window – IA –



Previsões utilizando Inteligência Artificial - IA - Deep Learning



RNN02 – LSTM Vanilla IA – Validação

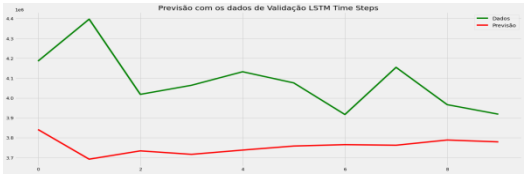
Epoch = 130 - evita o Overfitting

RNN02 – LSTM Vanilla IA – Validação

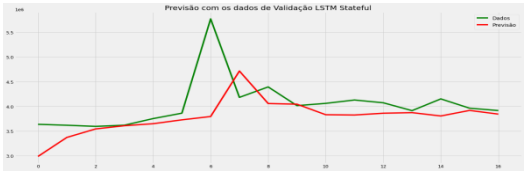
Epoch = 200

RNN02 – LSTM Window IA– Validação

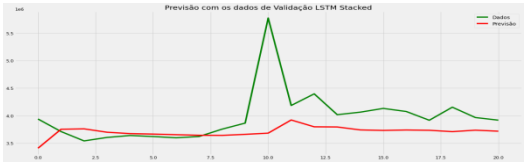
RNN02– LSTM Time Steps IA Validação



RNN02 – LSTM Stateful IA – Validação

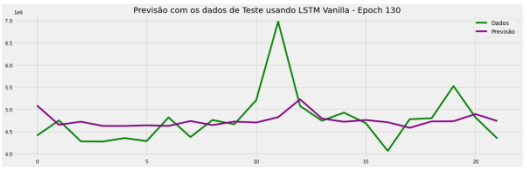


RNN02 – LSTM Stacked IA – Validação



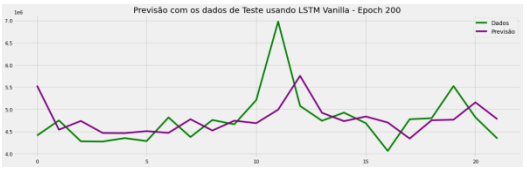
RNN02 – LSTM Vanilla IA – Teste

Epoch = 130 - evita o Overfitting



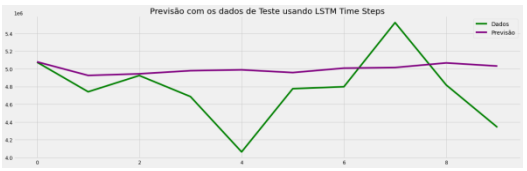
RNN02 – LSTM Vanilla IA – Teste

Epoch = 200

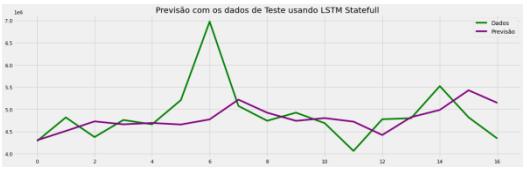


RNN02 – LSTM Window IA – Teste

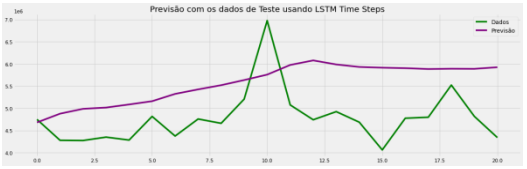
RNN02 – LSTM Time Steps IA – Teste



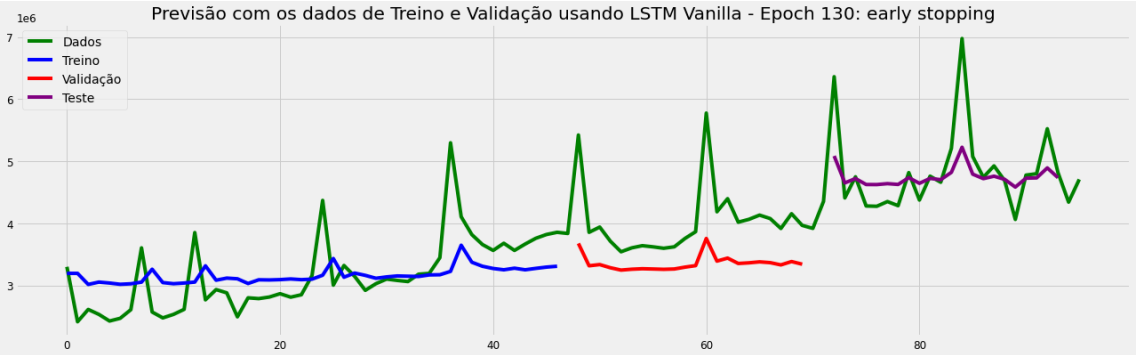
RNN02 – LSTM Stateful IA – Teste



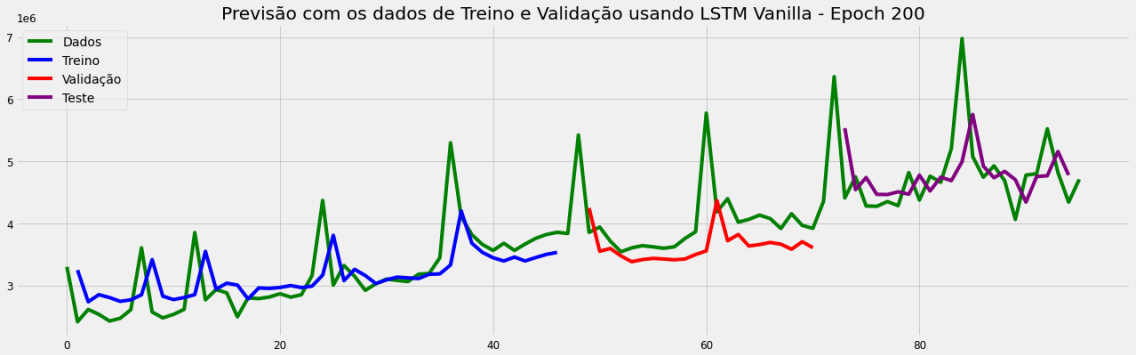
RNN02 – LSTM Stacked IA – Teste



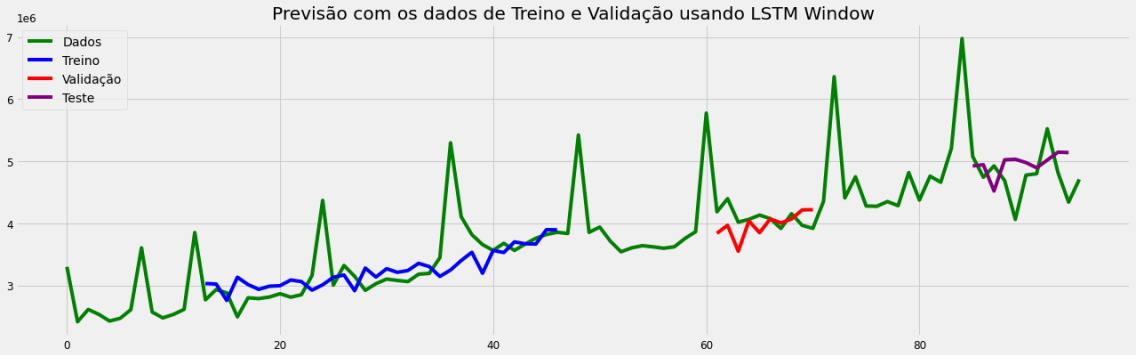
Modelo RNN02 – LSTM Vanilla IA – Epoch = 130 – evita o Overfitting



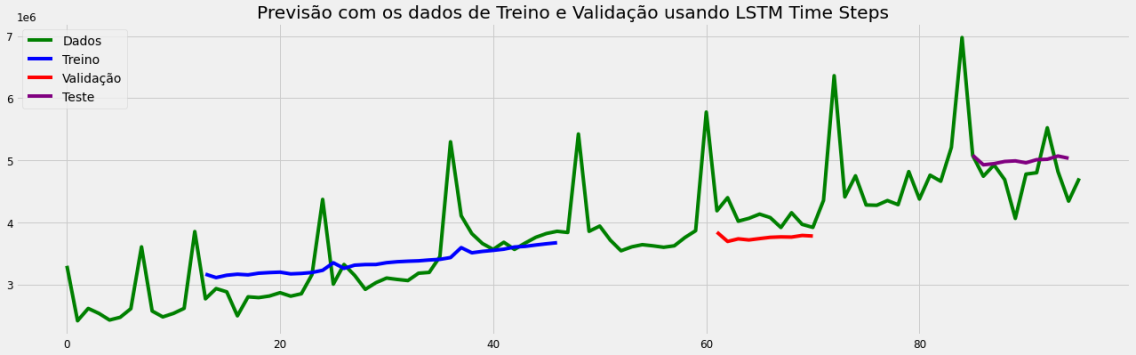
Modelo RNN02 – LSTM Vanilla IA – Epoch = 200



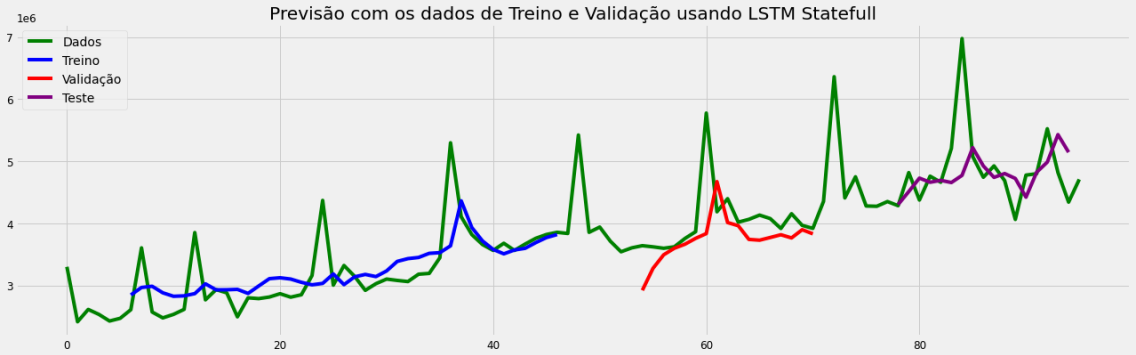
Modelo RNN02 – LSTM Window IA



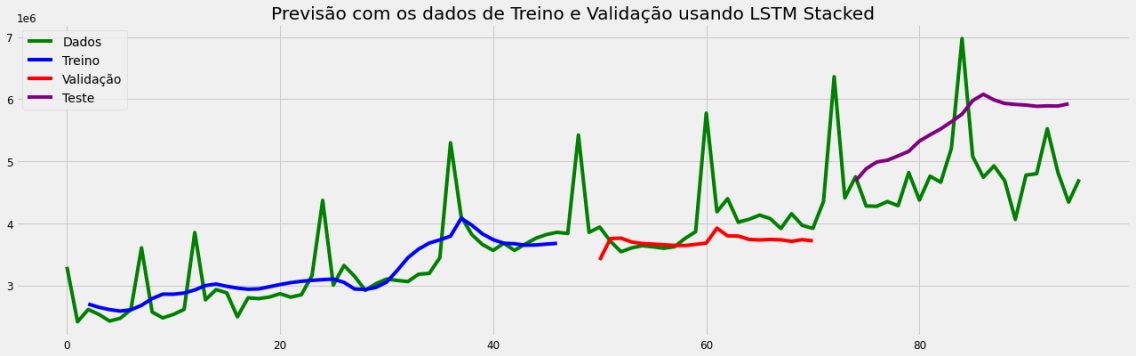
Modelo RNN02– LSTM Time Steps



Modelo RNN02– LSTM Stateful

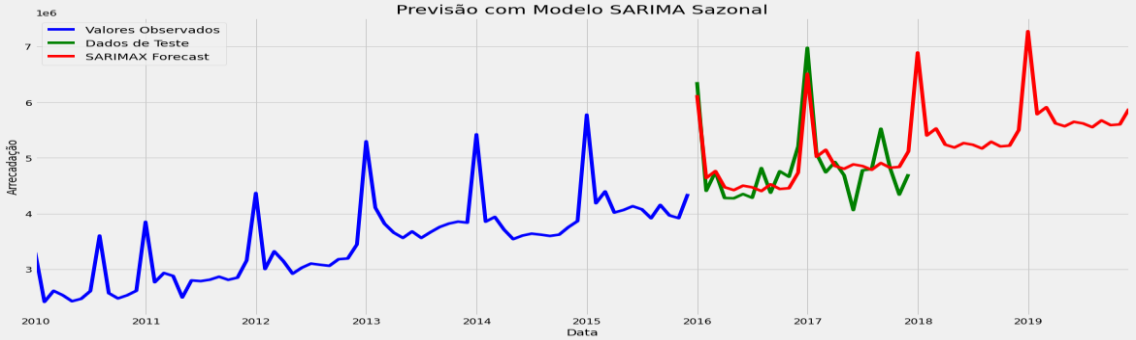


Modelo RNN02– LSTM Stacked

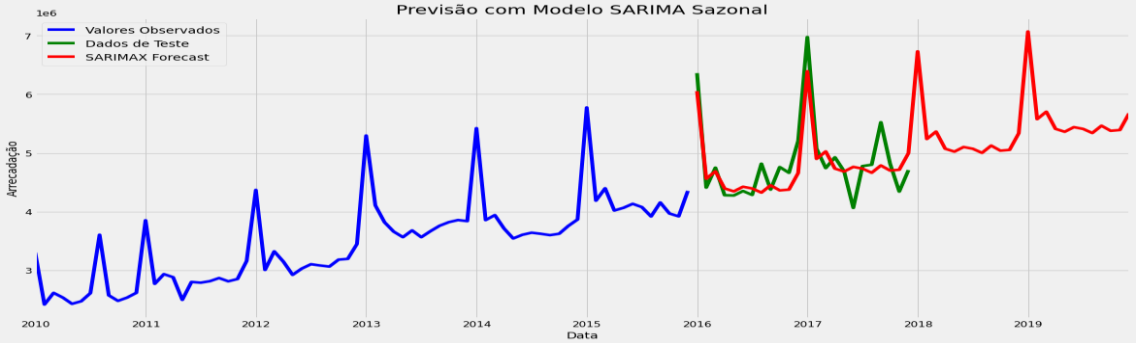


Forecasting utilizando Inteligência Artificial - IA

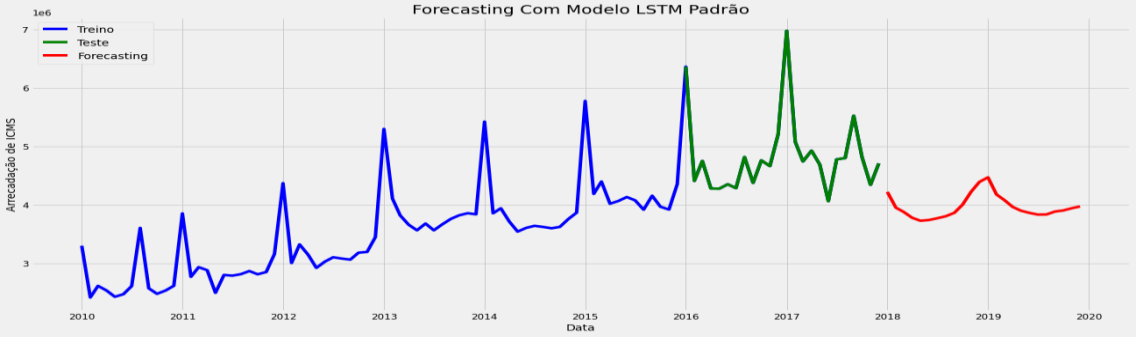
Modelo 17 – SARIMAX (0,1,0)(0,1,1,12)



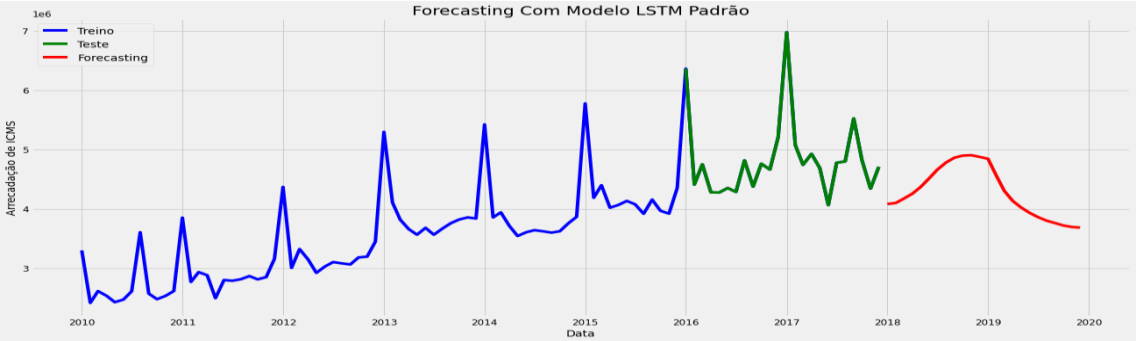
Modelo 18 – SARIMAX (1,0,1)(1,1,1,12)



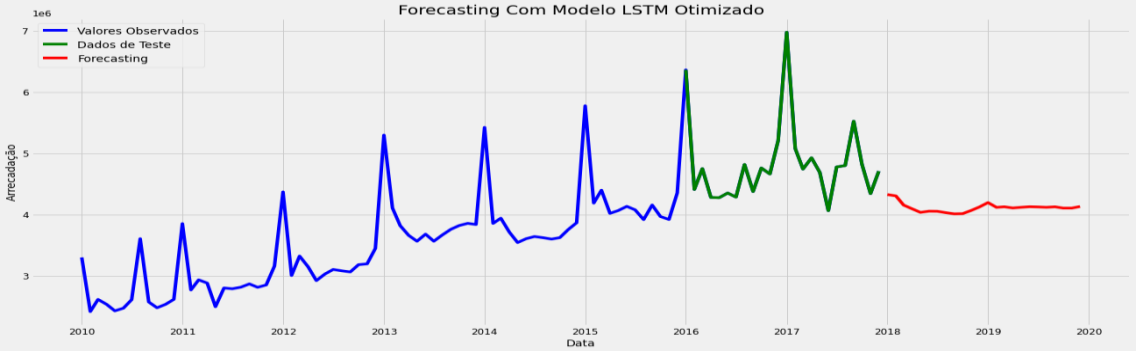
Modelo 22 – LSTM - IA (5 repetições)



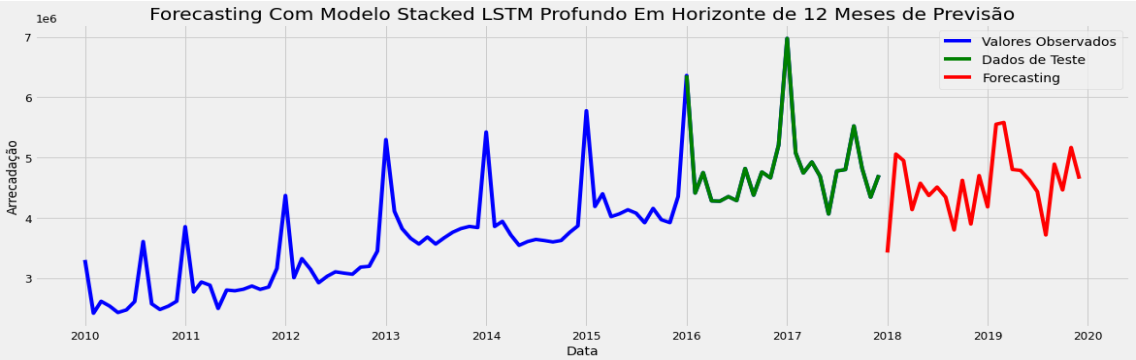
Modelo 22 – LSTM - IA (20 repetições)



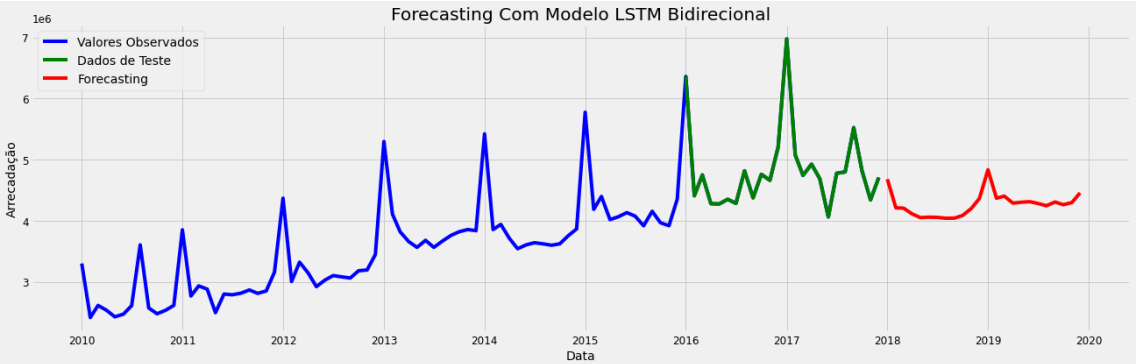
Modelo 23 – LSTM otimizado - IA



Modelo 24 – Stacked LSTM - IA



Modelo 25 – LSTM Bidirecional – IA



Resultados dos Modelos:

				RSME	RSME	RSME
Nº	Arquitetura	Modelo	Setup	treino	validação	teste
1	Métodos Estatísticos	BASE 11	Método Naive - p/ Mi			0,8054
2	Métodos Estatísticos	BASE 11	Método Naive - LOG			0,1551
3	Métodos Estatísticos	BASE 11	Método Naive			805471,0878
4	Métodos Estatísticos	Forecasting 12	Exponential Smoothing v1			857879,2406
5	Métodos Estatísticos	Forecasting 12	Exponential Smoothing v2			750397,7727
6	Métodos Estatísticos	ARIMA 13	ARIMA LOG (2, 1, 0)			767631,9506
7	Métodos Estatísticos	ARIMA 14	ARIMA LOG (1, 1, 1)			694827,8110
8	Métodos Estatísticos	ARMA 15	ARMA (1, 0)			1445179,5400
9	Métodos Estatísticos	ARMA 15	ARMA (5, 5)			1101078,9110
10	Métodos Estatísticos	ARMA 15	ARMA (12, 9)			706415,3914
11	Métodos Estatísticos	ARIMA 16	ARIMA (2, 0, 1)			1099391,396
12	Métodos Estatísticos	SARIMAX 17	SARIMAX (0, 1, 0) (0, 1, 1, 12)			332666,2626
13	Métodos Estatísticos	SARIMAX 18	SARIMAX (0, 1, 1) (0, 1, 1, 12)			336782,4202
14	IA - Deep Learning	LSTM 22	LSTM (5 repetições)			1037107,009
15	IA - Deep Learning	LSTM 22	LSTM (20 repetições)			878868,4191
16	IA - Deep Learning	LSTM 23	LSTM Otimizado			945778,1667
17	IA - Deep Learning	LSTM 24	Stacked LSTM			877490,6632
18	IA - Deep Learning	LSTM 25	LSTM Bidirecional			850832,6951
19	IA - Deep Learning	DeepAR 26	DeepAR			4874330,639
20	Inteligência Artificial - IA	RNA - MLP 1	MLP Vanilla	610423,2630		795393,9775
21	Inteligência Artificial - IA	RNA - MLP 2	MLP e Método Window	480295,7828		605733,0842
22	IA - Deep Learning	RNN - LSTM 1	LSTM Vanilla - 130 epochs	546148,5770	785606,6766	1417809,7913
23	IA - Deep Learning	RNN - LSTM 1	LSTM Vanilla - 200 epochs	496556,0484	586356,0493	950176,2755
24	IA - Deep Learning	RNN - LSTM 2	LSTM e Método Window	488946,9051	289236,3698	448340,4022
25	IA - Deep Learning	RNN - LSTM 3	LSTM e Time Steps	351787,0114	1184513,5124	1721690,0960
26	IA - Deep Learning	RNN - LSTM 4	LSTM e Stateful	444933,0519	549877,8552	911738,8182
27	IA - Deep Learning	RNN - LSTM 5	LSTM e Stacked	417403,4958	500538,7458	992971,6186

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 = 20
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 = 200)
```

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, epochs 5000, 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

Modelo RNN01 – 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')
for i 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')
for i 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