Forecasting - Previsões Para Séries Temporais

PipaFinder - Capacidade Mainframe

Guttenberg Ferreira Passos

O presente artigo tem como objetivo avaliar os modelos de inteligência artificial, aplicados a uma série temporal, e identificar o melhor resultado.

Os dados foram obtidos do monitoramento em curto prazo do consumo de MSUs do Mainframe, LPAR de Produção. Há também a necessidade de efetuar um planejamento de capacidade em longo prazo do crescimento do consumo dos “ofensores” do Mainframe.

Todos os modelos foram adaptados das aulas do curso de Modelagem Preditiva de Séries Temporais que podem ser encontradas na time line da Comunidade da Data Science Academy no portal: 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 usam-se Métodos Estatísticos, Inteligência Artificial, Machine Learning e Deep Learning, tais como os seguintes algoritmos:
  - Naive;
  - ARMA - Autoregressive Moving Average;
  - ARIMA - Autoregressive Integrated Moving Average;
  - SARIMAX – Seasonal Autoregressive Integrated Moving Average;
  - MLP - Multilayer Perceptron;
  - 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).

Naive significa ingênuo, portanto, não há técnica avançada aqui é apenas usada como ponto de partida, uma referência para os demais modelos. Qualquer modelo mais avançado deve apresentar resultados superiores ao Algoritmo Naive.

Smoothing

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 (Simple Moving Average).

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 (Moving Average Smoothing) é uma técnica eficaz na previsão de séries temporais também, ou seja, pode ser usada 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 Regressive Moving Average - 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

Auto Regressive Integrated Moving Average - ARIMA é 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

Seasonal Auto Regressive Integrated Moving Average - SARIMAX é um modelo sazonal de média móvel integrada auto-regressiva com 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 autoregressivo, diferenciado, e termos da média móvel da parte sazonal do modelo ARIMA.

Algoritmo MLP

Uma rede padrão Multilayer Perceptron - MLP é uma rede neural recorrente - RNN projetada 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

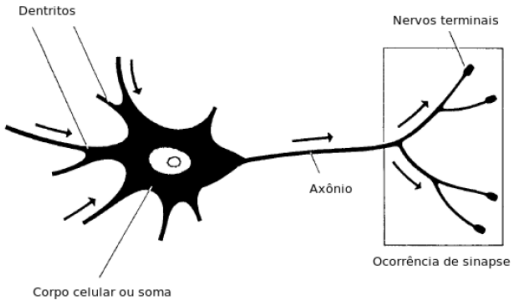
A rede Long Short Term Memory - LSTM é uma rede neural recorrente treinada usando Backpropagation Through Time e supera o problema da dissipação do gradiente. Como tal, o modelo pode ser usado para criar grandes redes recorrentes que, por sua vez, podem ser usadas para resolver problemas de sequência 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, 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.

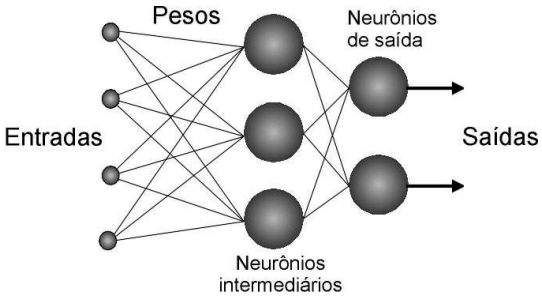
De acordo o com o Deep Learning Book, o neurônio biológico é uma célula, que pode ser dividida em três seções: o corpo da célula, os dendritos e o axônio, cada uma com funções específicas, porém complementares.



Fonte: Data Science Academy - DSA

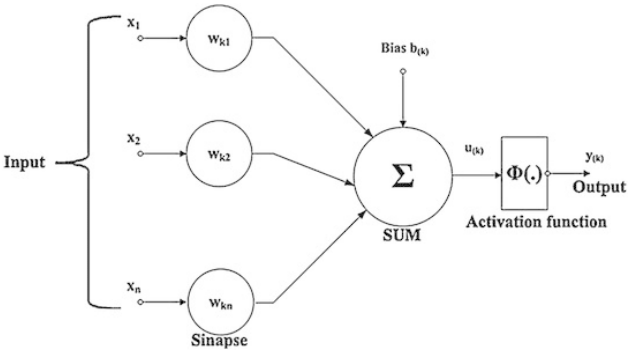
Neurônio Artificial

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

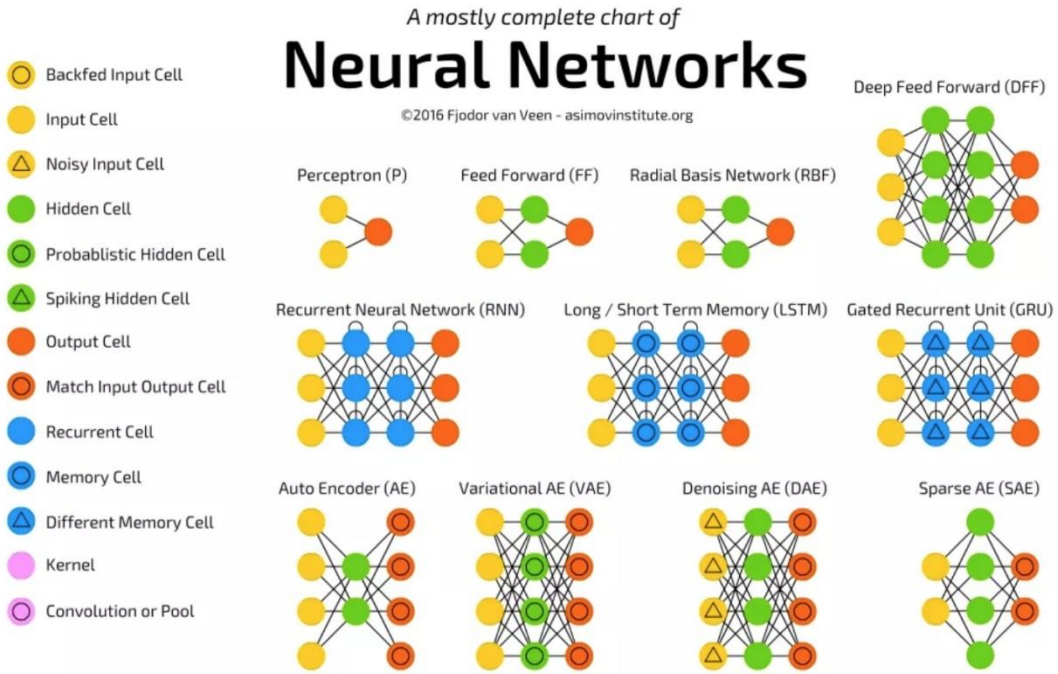
No aprendizado de máquina, o Perceptron é um algoritmo de aprendizado supervisionado de classificadores binários. Um classificador binário é uma função que pode decidir se uma entrada, representada por um vetor de números, pertence ou não a alguma classe específica. É um tipo de classificador linear, 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:

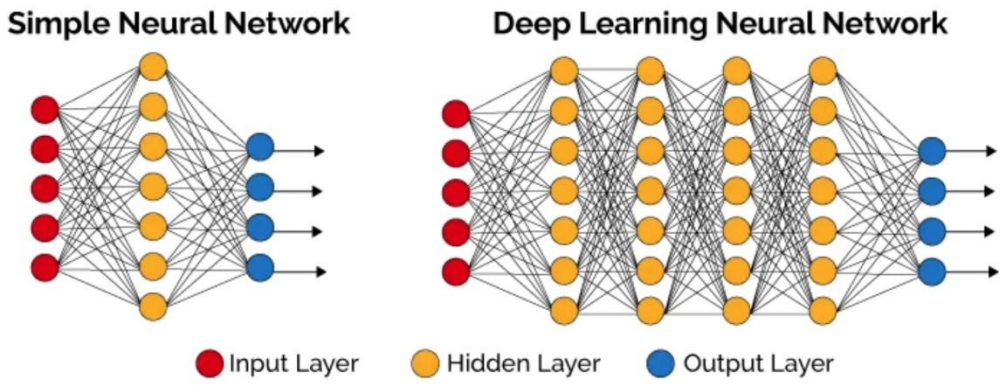


Fig4 – Rede Neural Simples e Rede Neural Profunda (Deep Learning)

Modelo Base

Definição do Problema de Negócio

A Prodemge possui um contrato com a IBM para utilização do Mainframe Z14, nesse contrato a Prodemge paga à IBM por faixa de consumo de MSU, o que passar de um determinado valor o custo aumenta em função da mudança de faixa de consumo.

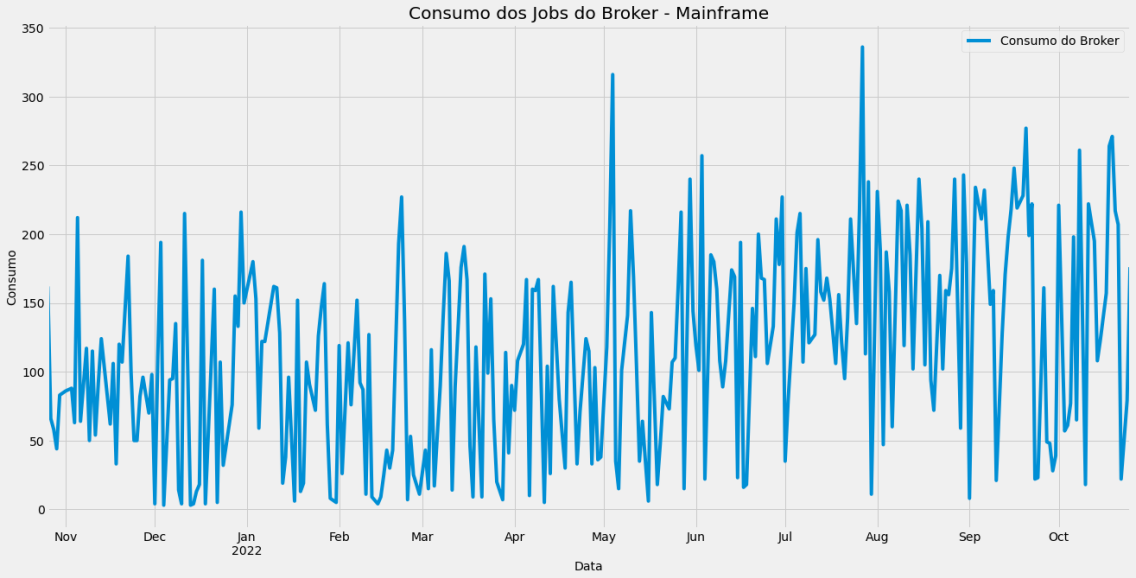
Há a necessidade de monitoramento em curto prazo do consumo de MSUs do Mainframe em horas, LPAR de Produção, para evitar o referido aumento.

Há também a necessidade de efetuar um planejamento de capacidade em longo prazo do crescimento do consumo dos “ofensores” do Mainframe.

Conjunto de Dados

Usaremos os conjuntos de dados obtidos do Mongo DB, banco de dados NoSql, que contêm o consumo de MSUs do Mainframe, para cada Broker, Service Name, Programa e Data. Os dados têm registros dos anos de 2021 e 2022.

Para comparação dos resultados de uma mesma amostra, será aplicado o seguinte filtro nos dados: Broker = ETB170, Service Name = PORTALDETRAN, Programa = NSDGX500.



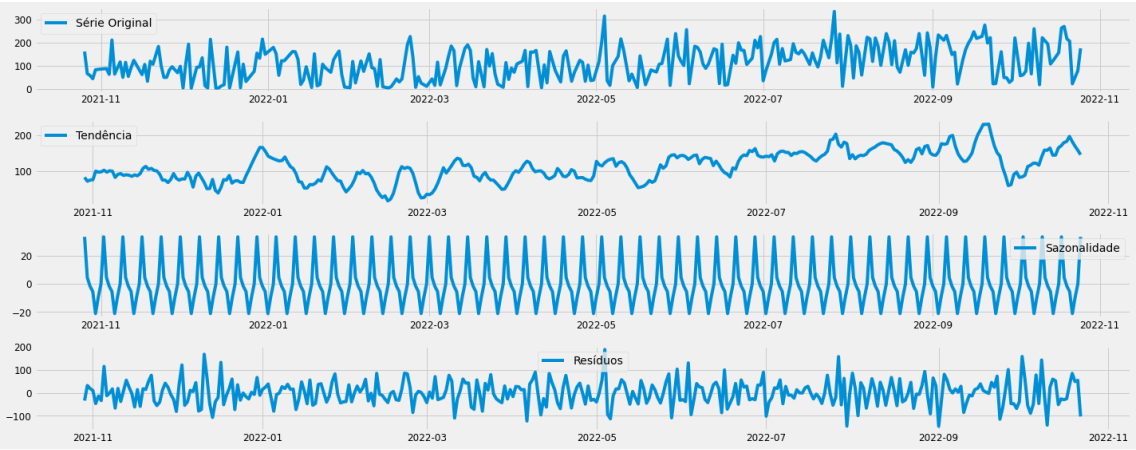
Análise Exploratória dos Dados

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:



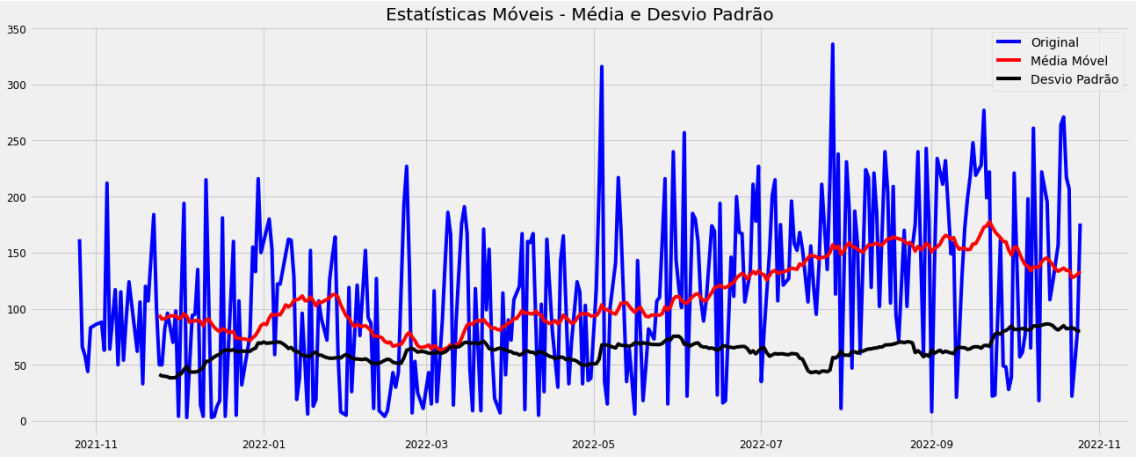
O gráfico acima mostra 3 componentes da série: Tendência, Sazonalidade e Resíduos.

- Tendência - ao longo do tempo, a série segue uma tendência de crescimento.
- Sazonalidade - o fenômeno se repete em períodos fixos.
- Resíduos - é o que sobra após a retirada da tendência e sazonalidade da série.

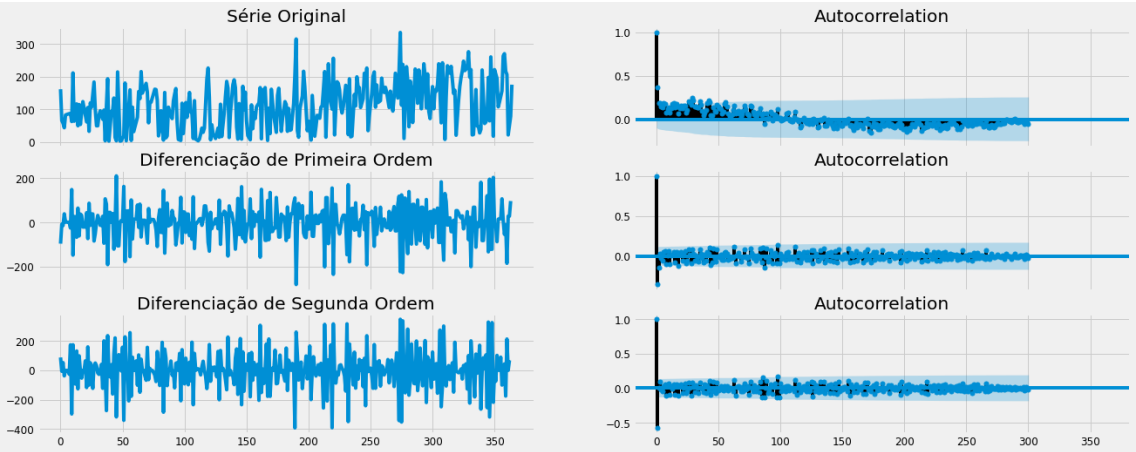
Estacionaridade da série.

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.



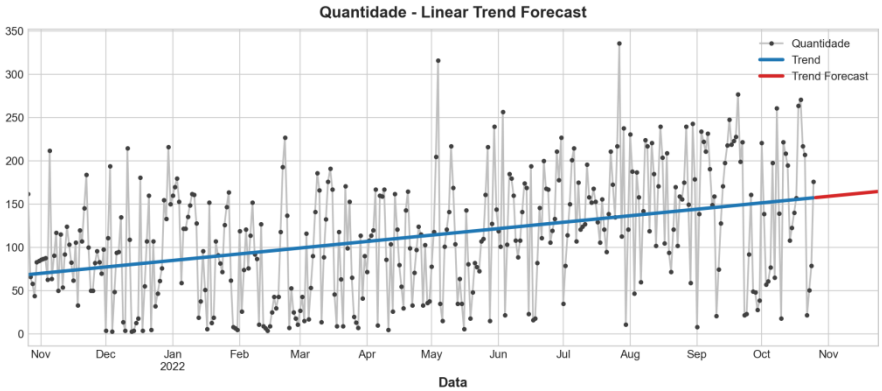
O gráfico ACF permite a avaliação da diferenciação mínima necessária para obter uma série estacionária (Parâmetro d para o Modelo ARIMA):



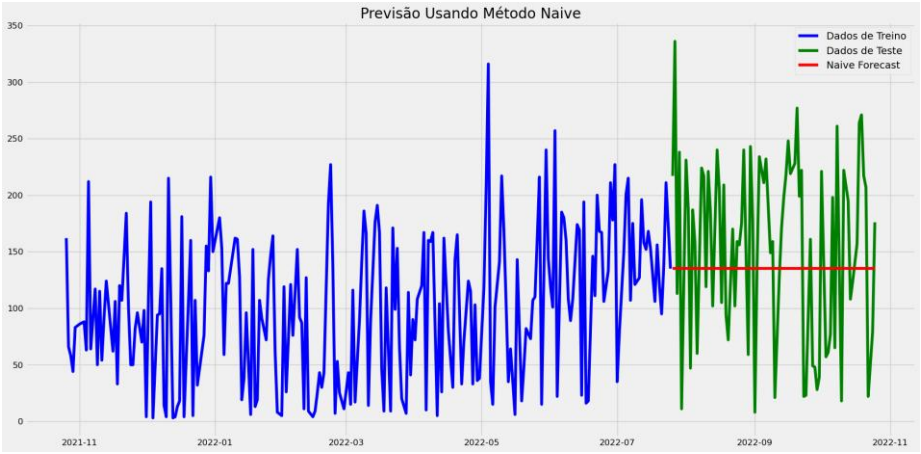
Para a execução dos modelos utilizando **Métodos Estatísticos** os dados foram separados em:

- 273 registros de treino (75%) e
- 92 registros de teste (25%).

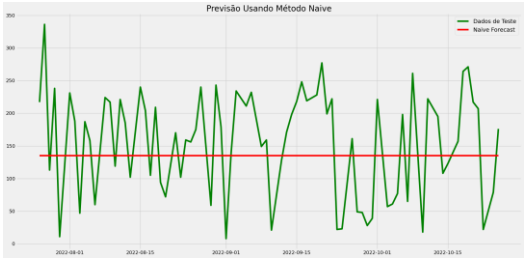
Modelo 01 – Previsões Método de Regressão Linear Simples MRLS



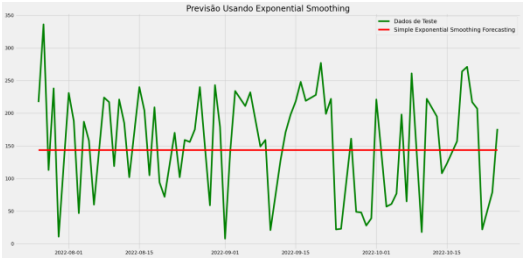
Modelo 11 – Previsões Método Naive



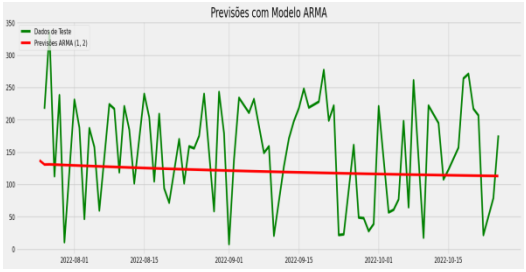
Modelo 11 – Método Naive



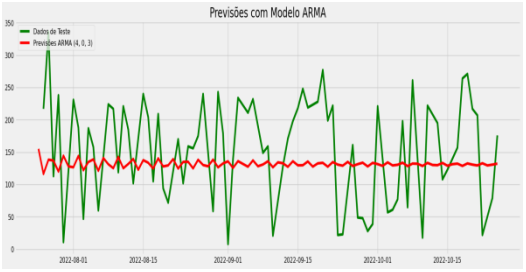
Modelo 12 – Smooting



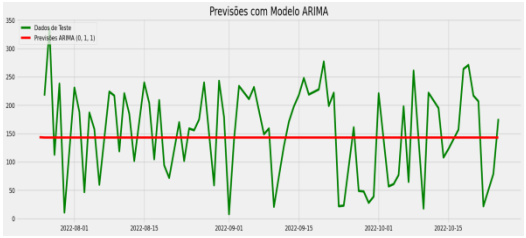
Modelo 15 – ARMA (1, 2)



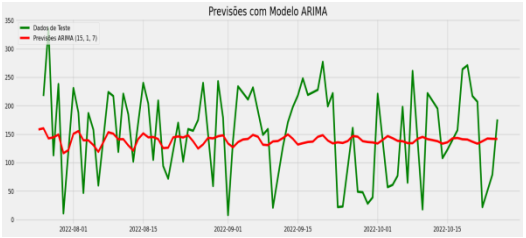
Modelo 15 – ARMA (4, 3)



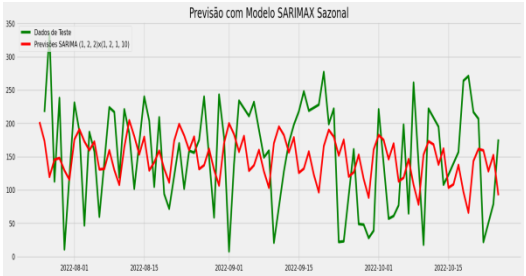
Modelo 16 – ARIMA (0, 1, 1)



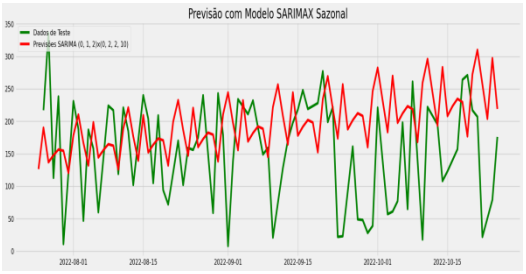
Modelo 16 – ARIMA (15, 1, 7)



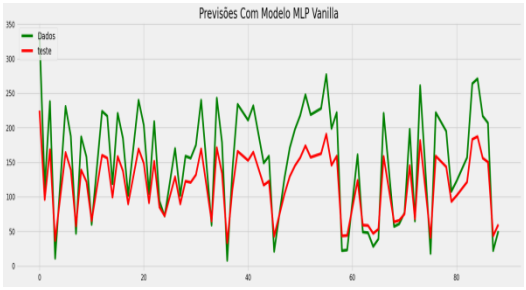
Modelo 17 – SARIMAX (1,2,2)(1,2,1,10)



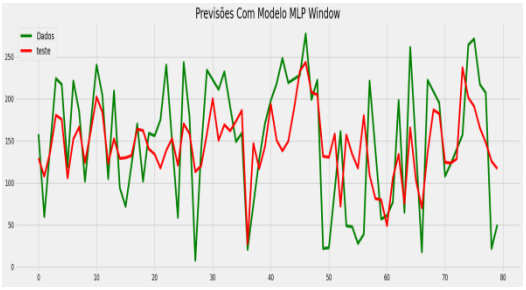
Modelo 18 – SARIMAX (2,2,0)(2,2,0,10)



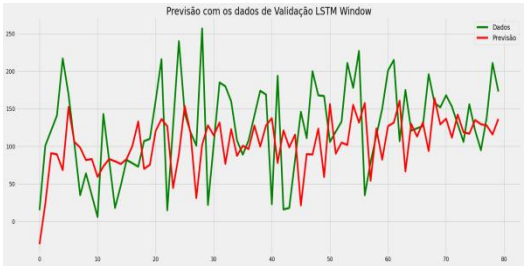
Modelo RNN01 – MLP Vanilla – IA



Modelo RNN01 – MLP Window – IA



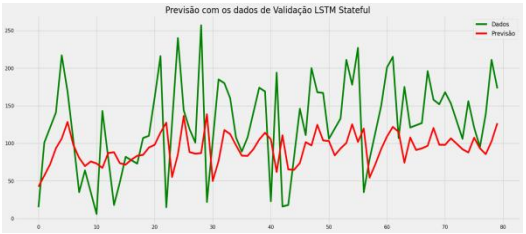
Modelo RNN02 – LSTM Window – IA



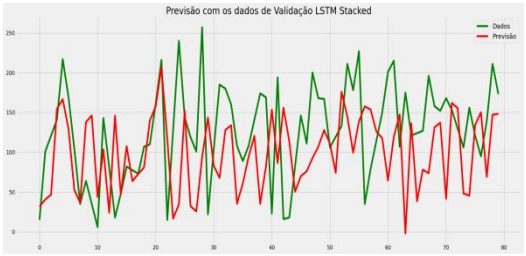
Modelo RNN02 – LSTM 130 early – IA

Modelo RNN02 – LSTM Time Steps – IA

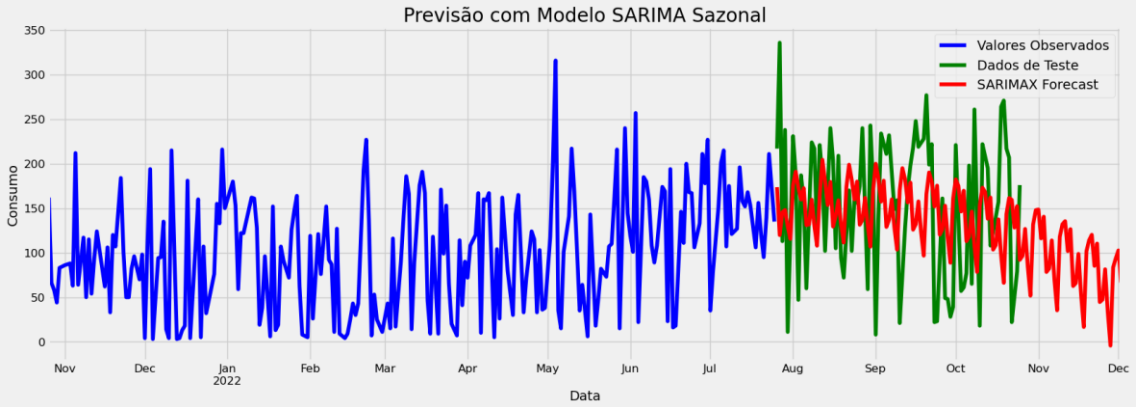
Modelo RNN02 – LSTM Stateful – IA



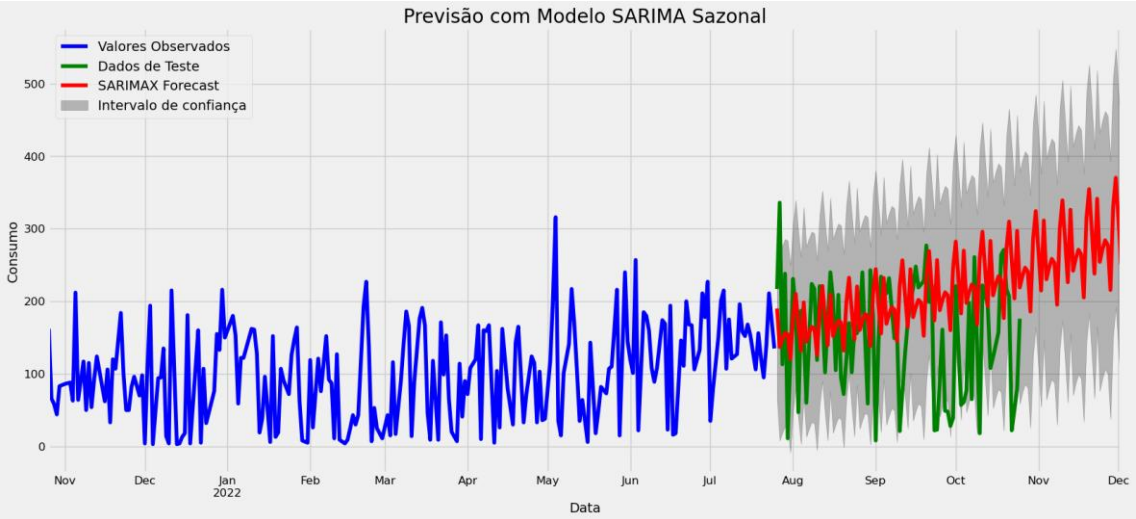
Modelo RNN02 – LSTM Stateful – IA



Forecasting utilizando Modelo 17 – SARIMAX (1,2,2)(1,2,1,10)

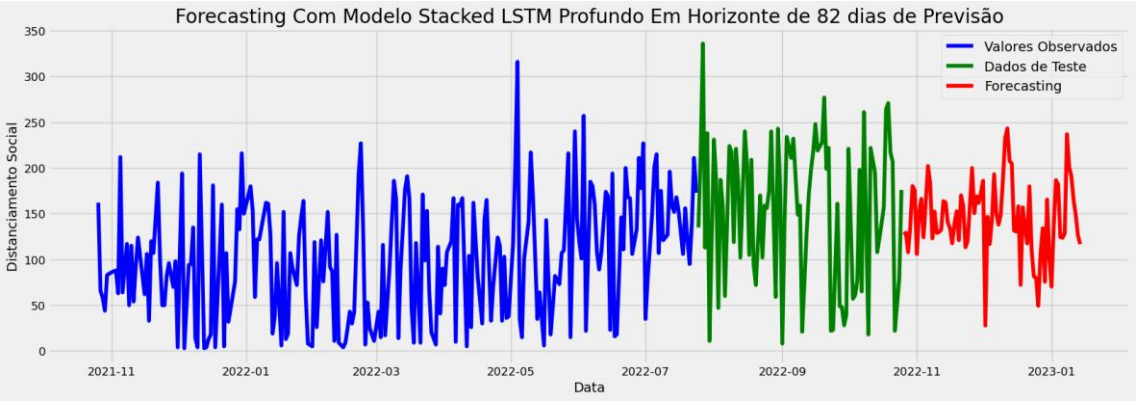


Forecasting utilizando Modelo 18 – SARIMAX (2,2,0)(2,2,0,10)

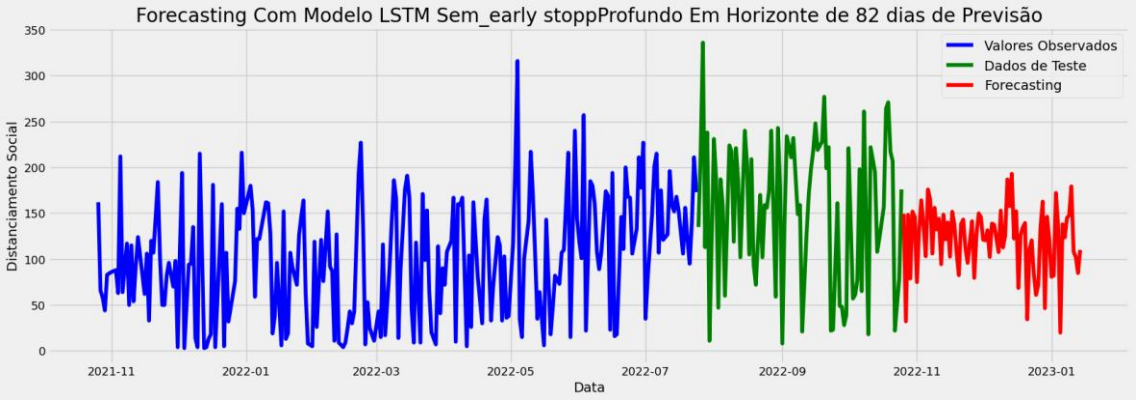


Forecasting utilizando IA Deep Learning

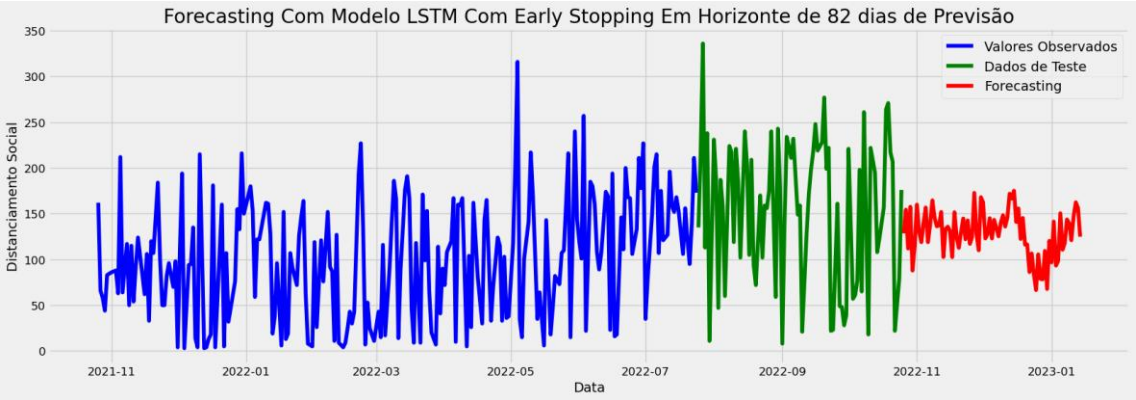
Modelo RNN01 – MLP Window – IA



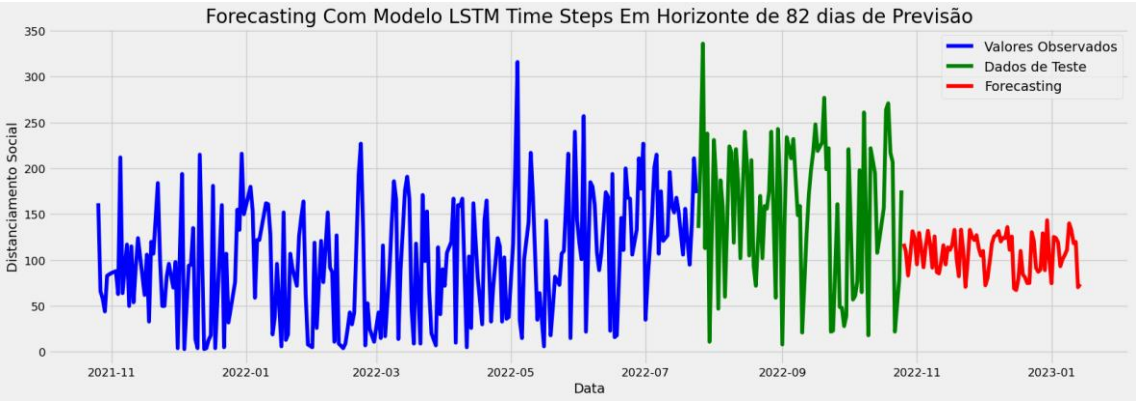
Modelo RNN02 – LSTM Window – IA



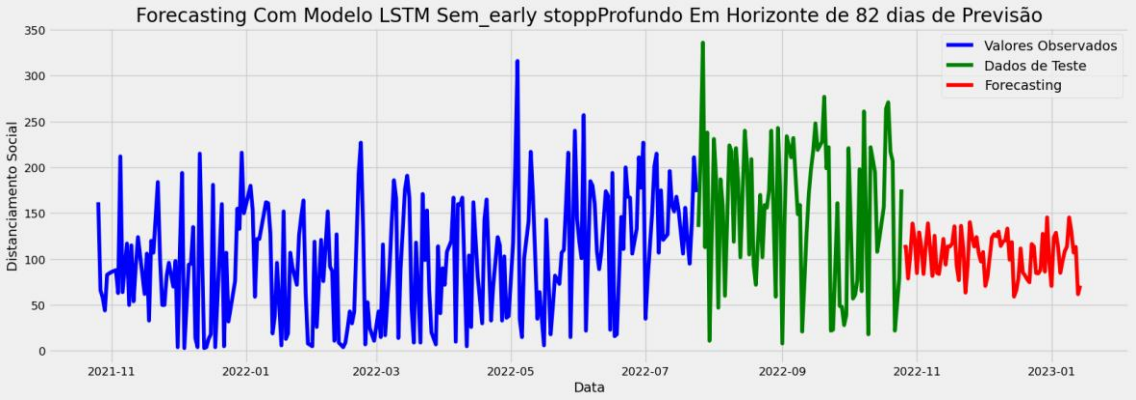
Modelo RNN02 – LSTM 130 early – IA



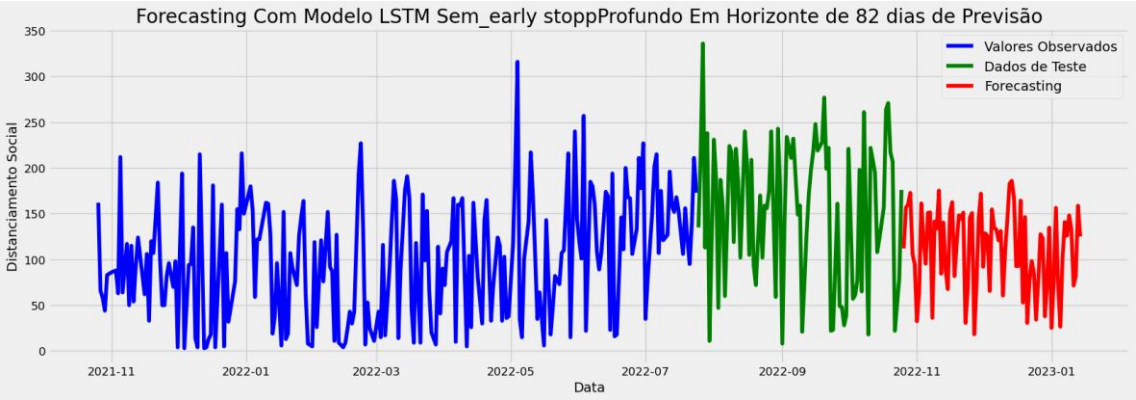
Modelo RNN02 – LSTM Time Steps – IA



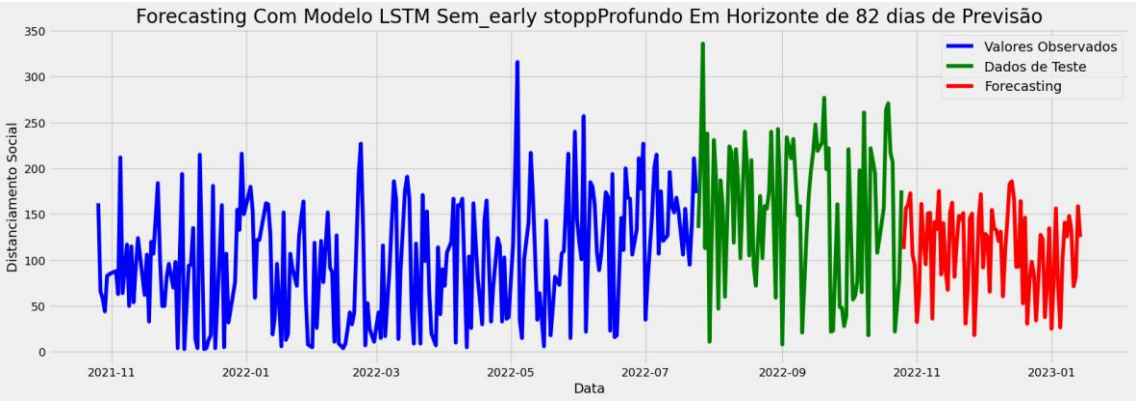
Modelo RNN02 – LSTM Stateful – IA



Modelo RNN02 – LSTM Stateful – IA



Modelo RNN02 – LSTM Stateful – IA



Resultados dos Modelos:

Previsão de Consumo de MSUs do Mainframe				RMSE	RMSE	RMSE
Nº	Arquitetura	Modelo	Setup	treino	validação	teste
1	Métodos Estatísticos	BASE 11	Método Naive			77,7967
2	Métodos Estatísticos	Forecasting 12	Exponential Smoothing v1			76,2817
3	Métodos Estatísticos	Forecasting 12	Exponential Smoothing v2			82,4853
4	Métodos Estatísticos	ARMA 15	ARMA (1, 2)			81,8648
5	Métodos Estatísticos	ARMA 15	ARMA (4, 3)			78,9103
6	Métodos Estatísticos	ARIMA 16	ARIMA (0, 1, 1) Previsão			76,3238
7	Métodos Estatísticos	ARIMA 16	ARIMA (15, 1, 7) Previsão			74,9877
8	Métodos Estatísticos	SARIMAX 17	SARIMAX (1, 2, 2) (1, 2, 1, 10)			67,0580
9	Métodos Estatísticos	SARIMAX 18	SARIMAX (0, 1, 2) (0, 2, 2, 10)			8,6038
10	Inteligência Artificial - IA	RNA - MLP 1	MLP Vanilla	62,3155		84,0842
11	Inteligência Artificial - IA	RNA - MLP 2	MLP e Método Window	53,3053		72,5207
12	IA - Deep Learning	RNN - LSTM 2	LSTM e Método Window	48,7170		67,1578
13	IA - Deep Learning	RNN - LSTM 1	LSTM e 130 epochs	62,865		63,1377
14	IA - Deep Learning	RNN - LSTM 3	LSTM e Time Steps	55,1184		65,8831
15	IA - Deep Learning	RNN - LSTM 4	LSTM e Stateful	53,4697		66,2286
16	IA - Deep Learning	RNN - LSTM 5	LSTM e Stacked	2,1089		78,0692
17	IA - Deep Learning	RNN - LSTM 2	LSTM e Método Window	48,7170	67,1578	85,3992
18	IA - Deep Learning	RNN - LSTM 1	LSTM e 130 epochs	62,865	63,1377	80,0989
19	IA - Deep Learning	RNN - LSTM 3	LSTM e Time Steps	55,3229	65,7431	82,1498
20	IA - Deep Learning	RNN - LSTM 4	LSTM e Stateful	53,4497	66,2286	83,3471
21	IA - Deep Learning	RNN - LSTM 5	LSTM e Stacked	2,1089	78,0692	97,5789

Arquiteturas dos Modelos utilizando IA Deep Learning

Modelo RNN01 – MLP Vanilla – IA

```
model = Sequential()
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 RNN02 – LSTM Window – IA

```
model = Sequential()
model.add(LSTM(4, input_shape = (1, look_back)))
model.add(Dense(1))
model.compile(loss = 'mean_squared_error', optimizer = 'adam')
monitor = EarlyStopping(monitor = 'val_loss', min_delta = 1e-4, patience = 5, verbose = 1, mode = 'auto')
model.fit(trainX, trainY, epochs = 200, batch_size = 1, verbose = 2)
```

Modelo RNN02 – LSTM Early– IA

```
model = Sequential()
model.add(LSTM(4, input_shape = (1, look_back)))
model.add(Dense(1))
model.compile(loss = 'mean_squared_error', optimizer = 'adam')
monitor = EarlyStopping(monitor = 'val_loss', min_delta = 1e-3, patience = 5, verbose = 1, mode = 'auto')
model.fit(trainX, trainY, validation_data = (validX, validY), callbacks = [monitor], verbose = 2, epochs = 200)
```

Modelo RNN02 – LSTM Time Steps – IA

```
model = Sequential()
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

```
batch_size = 1
model = Sequential()
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

```
batch_size = 1
model = Sequential()
model.add(LSTM(100, batch_input_shape = (batch_size, look_back, 1), activation = 'tanh', return_sequences =
True))
model.add(LSTM(50, activation = 'tanh', return_sequences = True))
model.add(LSTM(20,  activation = 'tanh'))
model.add(Dense(10, activation = 'tanh'))
model.add(Dense(5, activation = 'tanh'))
model.add(Dense(1))
model.compile(loss = 'mean_squared_error', optimizer = 'adam')
for i in range(300):
   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