UFC

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS JARDINS DE ANITA - ITAPAJÉ

Reconhecendo Padrões em Séries Temporais com Long Short-Term Memory

Giovanna Dias Castro de Oliveira
Jônatas Fernandes Silva
Leandro Nascimento Adegas

Disciplina: Aprendizado de Máquina e Reconhecimento de Padrões

Itapajé, Julho, 2025

Conteúdo

1 Introdução **3**

2 Background e Trabalhos Relacionados **3**

3 Modelo de Solução **5**
3.1 *Long Short-Term Memory* 5

4 Metodologia **5**
4.1 O Ambiente e os Datasets 5
4.2 Preparação e Tranformação dos Dados 6
4.3 O Código e as Condições de Execução dos Testes 6

5 Resultados **6**
5.1 Dataset Vendas de Shampoo 6
5.2 Dataset Vendas de Bicicleta 7

6 Conclusão **8**
6.1 Contribuição 8
6.2 Limitações e Trabalhos Futuros 9

7 Agradecimentos **9**

8 Referências **10**

Resumo

Este trabalho trata da experimentação e comparação do comportamento de dois modelos de redes neurais em relação a análise e reconhecimento de padrões em séries temporais. Os modelos utilizados para isso foram: Memória de Longo Prazo (*long short-term memory*, LSTM). A realização do experimento foi divida em duas partes, que iremos chamr de parte 01 e parte 02, para efeitos de praticidade. A parte 01 foi realizada utilizando um dataset de vendas de *shampoo*, já a parte 02 utilizou o dataset de vendas de bicicletas. A principal contribuição deste trabalho se dá pelas observações realizadas com relação ao desempenho e padrões reconhecidos durante a experimentação. Em síntese, foi notado que, o modelo tem um bom desempenho, principalmente quando há muitos dados já que o LSTM tem memória de longa duração.

1 Introdução

Série temporal é uma coleção de pontos registrados no decorrer de um intervalo de tempo [1]. Como exemplo de série temporal podemos citar a variação nos valores de ativos de investimentos da bolsa de valores do Brasil (B3). Com o uso de séries temporais pode-se processar dados históricos que por sua vez poderão ser utilizados para prever os eventos futuros. Outro exemplo da análise e reconhecimento de padrões que estamos vivenciando atualmente é o aumento das explosões solares, que notoriamente ocorre a cada 11 anos [2], um padrão que pode ser identificado através da análise de séries temtporais.

Dado a importância das séries temporais, analisá-las de forma coerente deve ser um fator a se observar. A escolha de um bom classificador é uma parte importantíssima do processo de análise e reconhecimento de padrões. Para este trabalho, propõe-se a experimentação e utilização do LSTM, um modelo de classificação baseado em redes neurais. O LSTM é uma evolução das redes neurais recorrentes (*recurrent neural network, RNN* [3].

Para que a experimentação pudesse ser realizada, foram utilizados dois datasets já prontos, disponibilizados pelo professor Dr. Júlio César Santos dos Anjos, autor já citado acima. A saber, os datasets utilizados nas parte 01 e 02 deste trabalho foram um conjunto de dados de vendas de *shampoo* e um segundo conjunto de dados de vendas de bicicletas. A análise dos dados envolveu todas as etapas de transformação dos dados [4]. Ademais, o objetivo deste trabalho é explorar e analisar o comportamento do modelo LSTM quando se trata de análise e reconhecimento de padrões em séries temporais.

A organização deste trabalho segue a seguinte ordem. A seção 2 contém os fundamentos necessários para a realização dos experimentos. A seção 3 explica de maneira breve o funcionamento do LSTM, além de explicar o modelo de solução proposto neste artigo. A seção 4 explica o ambiente e os métodos utilizados para as demonstrações empíricas deste trabalho. A seção 5 traz os resultados obtidos após a aplicação do modelo de solução. Por fim, a seção 6 traz a conclusão, contribuição, limitações deste trabalho, além de possíveis trabalhos futuros.

2 Background e Trabalhos Relacionados

Para entender o propósito, objeto e o objetivo deste trabalho é necessário haver a compreensão de alguns aspectos e do contexto em que ele está inserido. Para que isso seja possível foram análisados, explorados e explicados abaixo de forma parcial alguns trabalhos que são considerados importantes para o bom entendimento do assunto. Na tabela 1 pode-se em uma visão top-down os trabalhos que tem relação com o presente artigo e que serão abordados nesta seção. É importante citar que, alguns dos trabalhos trazidos abaixo comparam RNNs com LSTM, o trazemos estas comparações devido ao LSTM ser uma variação de RNN [3], logo, se faz necessário a comparação, para que por meio desta se justifique o uso do LSTM ao invés de RNN neste trabalho.

Tabela 1: Referências Base para o Trabalho

Reconhecimento de Padrões – Análise de Séries Temporais
Recurrent Neural Networks: A Comprehensive Review of Architectures, Variants, and Applications
Performance Analysis of Long Short-Term Memory Predictive Neural Networks on Time Series Data
Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling
An Empirical Exploration of Recurrent Network Architectures
Deep Learning with Python
Long Short-Term Memory

Séries temporais representam uma sequência de observações coletadas ao longo do tempo, comumente utilizadas em áreas como finanças, varejo, e energia [1]. A análise dessas séries tem como principal objetivo prever comportamentos futuros a partir de padrões extraídos dos dados históricos. Com o crescimento da capacidade computacional e da disponibilidade de dados, técnicas de aprendizado de máquina tornaram-se ferramentas promissoras para esse tipo de análise.

As redes neurais recorrentes (RNNs) surgem como uma arquitetura fundamental para lidar com dados sequenciais devido à sua habilidade de manter informações contextuais por meio de um estado interno atualizado ao longo da sequência [5]. No entanto, RNNs tradicionais enfrentam dificuldades como o problema do gradiente desaparecendo, o que compromete sua performance em sequências longas [6]. Para contornar esse desafio, foram propostas arquiteturas aprimoradas, como a *Long Short-Term Memory* (LSTM), que introduz mecanismos de memória mais robustos, permitindo o aprendizado de dependências de longo prazo [7].

Estudos comparativos, como o de Chung et al. [8], demonstram que o LSTM supera as redes RNNs tradicionais em tarefas de modelagem sequencial. Além disso, análises como a de Jozefowicz et al. [6] exploram variantes estruturais da LSTM, ressaltando a importância de componentes como a "*forget gate*", cuja modificação (por exemplo, com viés adicionado) pode impactar positivamente o desempenho do modelo.

No contexto de séries temporais, aplicações práticas com LSTM têm sido eficazes em tarefas como previsão de vendas, monitoramento de processos industriais e detecção de anomalias [9]. Um ponto central nas pesquisas atuais envolve a escolha e otimização dos hiperparâmetros desses modelos, que influenciam diretamente na capacidade preditiva. Trabalhos empíricos, como o de Bolboacă e Haller [9], realizam análises sistemáticas variando parâmetros como o tamanho da janela de entrada, número de unidades ocultas e taxa de aprendizado, oferecendo contribuições valiosas para o uso prático de LSTM em predições.

O uso de LSTM em ambientes acadêmicos também tem sido tema de estudo em disciplinas relacionadas ao reconhecimento de padrões, como apresentado por Júlio Anjos [1], que introduz essa rede como ferramenta que tem grande valor para a identificação de comportamentos sazonais, cíclicos e anômalos em dados temporais. Por fim, obras de referência como a de Chollet [10] contribuem para a popularização e aplicação prática dessas arquiteturas, demonstrando com exemplos concretos como o modelo LSTM pode ser empregados eficientemente em tarefas de séries temporais usando frameworks modernos como o Keras e TensorFlow.

Em resumo, a literatura aponta que LSTM apresenta vantagens sobre RNN em tarefas que exigem memórias mais longas, e que a escolha do modelo ideal pode depender fortemente do tamanho do conjunto de dados, dos padrões envolvidos e dos objetivos preditivos. Este trabalho busca explorar e validar empiricamente esses aspectos por meio de experimentos com dois conjuntos de dados de vendas de diferentes nichos, oferecendo uma comparação prática entre os modelos mencionados.

3 Modelo de Solução

Nesta seção será apresentado como do LSTM funciona de breve maneira, para que se possa ter uma visão de como é realizado todo o processo por trás dos panos, assim compreendendo-o de forma clara. Além disso, será explicado o modelo de solução sugerida dado o problema proposto: analisar e reconhecer padrões em uma série temporal através do LSTM, esclarecendo como ele será aplicado para resolver o problema em questão.

3.1 *Long Short-Term Memory*

A rede LSTM é um modelo de rede neural idealizado por Hochreiter e Schmidhuber em 1997 [7]. Surgiu com o objetivo de superar a limitações como a do desaparecimento de gradiente durante o treinamento. Sua arquitetura possui células de memória de longa duração que mantém informações ao longo do tempo e por meio de mecanismos de controle, a saber, portas de entrada e saída, que determinam, respectivamente, quais dados devem ser armazenados, descartados ou repassados para a próxima etapa da rede [7], como demonstrado na Figura 1. Neste trabalho, o LSTM foi utilizado para a análise de duas séries temporais compostas por dados multivariados, o que permitiu a identificação e o reconhecimento de padrões. O modelo utilizou um dado histórico anterior para prever o próximo valor, ou seja, a previsão é igual a $i + 1$.

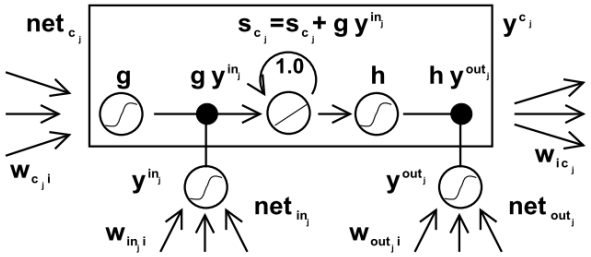


Figura 1: Arquitetura da Célula [7]

4 Metodologia

Nesta seção será apresentada a metodologia utilizada para a realização do experimento prático com o LSTM. Desde o ambiente utilizado para a realização dos testes, os datasets, as etapas de preparação e transformação dos dados até a execução do código em python.

4.1 O Ambiente e os Datasets

Para realizar a experimentação da análise e reconhecimento de padrões em séries temporais utilizando o LSTM, foi utilizado o ambiente do *Google Colab*. Plataforma online do *Google* que simula um *Jupyter notebook*. Os datasets utilizados foram: vendas de shampoo, em formato CSV, disponível em `https://drive.google.com/file/d/1Ucu9bpFpHIMFcq9M5aYWWTY8SSY_ZcoY/view?usp=drive_link`, e o dataset de vendas de bicicleta, que foi obtido através do *scikit-learn*, como descrito no código abaixo.:

Listing 1: Conjunto de Dados Venda de Bicicletas

```
from sklearn.datasets import fetch_openml

bike_sharing = fetch_openml("Bike_Sharing_Demand", version=2, as_frame=True)
df = bike_sharing.frame
```

4.2 Preparação e Tranformação dos Dados

Para a preparação e transformação dos dados foi realizando seguindo as intruções que tivemos desde a primira aula da disciplina de Aprendizado de Máquina e Reconhecimento de Padrões, sendo estas etapas todas descritas conforme a Figura 2. Dentro dos *notebooks* pode-se notar que as etapas foram descritas antes da execução de cada trecho de código. Pra que assim ficasse de explícito a responsabilidade de cada parte do código.

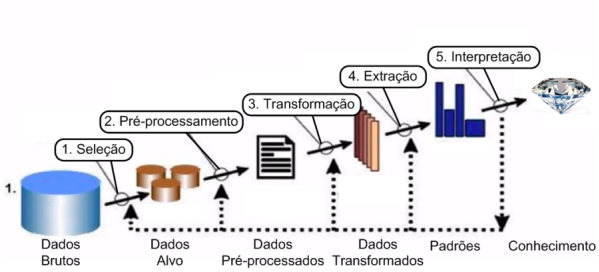


Figura 2: Estapas Preparação e Tranformação dos Dados Em Informação [4]

4.3 O Código e as Condições de Execução dos Testes

Os *notebooks* contento os códigos podem ser encontrados em `https://colab.research.google.com/drive/10TYzxdEr-DUN31NcSDSvhoJTMBoWMdnF?usp=sharing`, parte 01 da atividade, `https://colab.research.google.com/drive/1sZ33YTI8U0QXS7r44LUg_EwcRlICPplq?usp=sharing`, parte 02 da atividade, também se encontra disponível em `https://github.com/jonatasfernandessilva7/TL-05-Aprendizado-de-maquina-e-reconhecimento-de-padroes.git`, onde se encontram todos os códigos em um único repositório do *GitHub*.

A execução dos testes foi realizada nas seguintes condições. Em ambos os datasets, o modelo foi treinado em várias épocas, porém, o número que consideramos ideal ao final dos experimentos foi 12 épocas para o dataset vendas de shampoo (parte 01) e 15 épocas para o dataset vendas de bicicleta (parte 02). O valor da camada "*Dense*"permaneceu em 1 e a camada "*neurons*"teve seu valor alterado durante os testes, mas, ao final, a melhor percepção foi com seu valor entre 2, 4 e 8. Algo que também pode alterar os resultados é o "*lag*", contudo, mantivemos ele com o valor igual a 01, todos estes valores são comuns as partes 01 e 02. A descrição completa da atividade se encontra no apêndice 7 "Descrição da Atividade".

5 Resultados

Nesta seção serão apresentados os resultados obtidos com a experimentação do LSTM em ambos os datasets. Primeiramente analisaremos os resultadoss encontrados no dataset de vendas de shampoo e em seguida analisaremos os resultados obtidos através dos testes realizados no dataset de venda de bicicletas.

5.1 Dataset Vendas de Shampoo

Primeiramente, o nosso código resultou em um gráfico diferente do proposto na descrição original, encontrada no apêncide "Descrição da atividade"7. Essa diferença se deu pelo simples motivo da diferença na versão das bibliotecas utilizadas no código original, na versão disponibilizada na seção 4, no código da parte 01, dataset de vendas de shampoo estão descritas as mudanças feitas para que o código pudesse funcionar corretamente e plotar o gráfico.

Notamos que por conta do tamanho do dataset, com muitas épocas e muitos neurônios, o modelo começava a "alucinar", de certa maneira. Então, por esse motivo resolvemos realizar os testes com poucas épocas e com a base de 4 neurônios, como descrito na seção 4. Em nossos testes, o erro quadrático médio (RMSE) foi de 112.793 e a perda ou *loss* foi igual a 0.1955 na última época. Na Figura 3, podemos ver que os valores previstos pelo modelo ficaram bem próximos dos esperados, porém, um pouco a frente, isto ocorre por conta do $i + 1$, citado anteriormente.

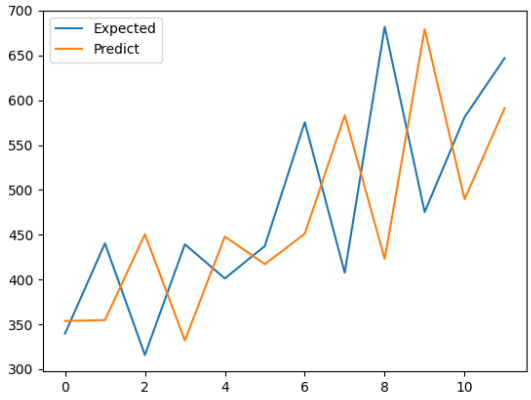


Figura 3: Resultados Dataset Vendas de Shampoo com 12 Épocas

Já na Figura 4, podemos ver o teste realizado com 06 épocas, porém, dessa vez os valores previstos pelo modelo ficaram bem mais ajustados em relação aos valores esperados. Esse fato ocorreu por uma pequena mudança realizada no *range* de plotagem, por sugestão do nosso nobre colega, Mário. Em vez de continuar usando a função *range*, utilizamos *np.arange*, como descrito abaixo, Listing 2. Para este teste o RMSE foi igual a 134.266 e a perda foi de 0.2361 na última época.

Listing 2: Alteração de Código

```
''' cdigo original '''
pyplot.plot(range(1, 13), expected_values, label='Expected')
pyplot.plot(range(1, 13), predictions, label='Predict')

''' cdigo alterado '''
pyplot.plot(range(1, 13), expected_values, label='Expected')
pyplot.plot(np.arange(0, 12), predictions, label='Predict')
```

Nos dois casos de testes notou-se que, quanto mais épocas mais as retas ficavam destorcidas, mesmo que um ponto ou outro ficasse próximos, nunca se tocavam, ou ficavam muito acima ou muito abaixo. Quando tentamos mudar o número de neurônios ocorreram quase o mesmo problema. Quando o número de neurônios era muito grande (24 foi o máximo que testamos) os valores ficaram muito distantes, apesar de o RMSE ficar igual a 97, em média. Ainda é importante citar que, ao observar os gráficos pode-se ver que os picos sempre ocorrem entre os meses de dezembro e julho, período de férias.

5.2 Dataset Vendas de Bicicleta

Para o dataset de vendas de bibicletas, o resultado da previsão (linha em amarelo), ficou bem parecido com o original que foi passado na descrição da atividade. Porém, em relação aos valores esperados no treinamento do modelo ficaram bem distorcidos. Contudo nota-se um padrão

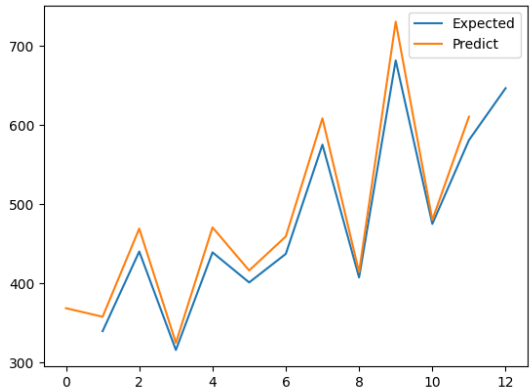


Figura 4: Resultados Dataset Vendas de Shampoo com 12 Épocas

recorrente nos dois cenários, entre os dias há um pico de vendas, e logo após uma caída brusca, formando uma espécie de montanha russa. No início e fim de cada dia a reta começa em baixa e ao longo do dia ela cresce, o que demonstra que o fluxo é intenso durante todo o dia, todos os dias, menos no sábado e domingo. Para além disso, destaca-se que o RMSE neste caso foi de 135.66 e a perda foi igual a 0.0084 na última época.

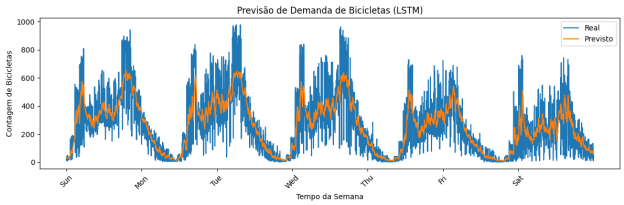


Figura 5: Resultados Dataset Vendas de Bicicleta

6 Conclusão

Este trabalho que teve como objetivo investigar e experimentar o uso do LSTM na análise e reconhecimento de padrões em séries temporais, utilizando dois conjuntos de dados diferentes: vendas de shampoo e vendas de bicicleta. Os experimentos que o LSTM é muito bom para prever valores com base em valores históricos anteriores. Nas subseções abaixo serão descritos a contribuição deste trabalho e as limitações que tivemos para realizá-lo.

6.1 Contribuição

A principal contribuição deste trabalho se deu pela aplicação prática e comparativa do modelo LSTM em dois conjuntos de dados distintos de séries temporais. O estudo demonstrou a sensibilidade do modelo a diferentes tamanhos de dataset, quantidade de épocas e número de neurônios, além de explorar a importância do ajuste de hiperparâmetros para a melhoria dos resultados.

Adicionalmente, o projeto proporcionou uma ótima experiência prática com a execução das etapas da análise de dados para este contexto, a saber, pré-processamento, modelagem, treinamento e avaliação de redes neurais recorrentes com memória de longa duração, oferecendo incentivos para decisões futuras em projetos de previsão temporal, como prever preços de investimentos na bolsa de valores.

6.2 Limitações e Trabalhos Futuros

Como limitações, destacam-se o uso de apenas dois datasets relativamente simples e o número restrito de variações testadas nos hiperparâmetros do modelo. Além disso, o LSTM, apesar de eficaz, pode não ser a melhor opção em todos os contextos, sobretudo em séries muito curtas ou com alta variabilidade não periódica. Para trabalhos futuros pensamos em avaliar o desempenho do LSTM em séries mais complexas, multivariadas e com maior volume de dados.

7 Agradecimentos

Agradecemos primeiramente a Deus pela inspiração e guia, ao professor Júlio César Santos dos Anjos pela orientação e dedicação, e aos valorosos colegas pela parceria e contribuições.

8 Referências

[1] J. C. S. dos Anjos, "Reconhecimento de padrões – análise de séries temporais." Aula ou apresentação acadêmica, 2025.

[2] "Ciclo solar." `https://pt.wikipedia.org/wiki/Ciclo_solar`. Online; Acesso em 2025-07-20.

[3] D. S. Academy, *Deep Learning Book*. Online, 2025. `https://www.deeplearningbook.com.br/arquitetura-de-redes-neurais-long-short-term-memory/`.

[4] J. C. S. dos Anjos, "Introdução ao aprendizado de máquina e reconhecimento de padrões." Aula ou apresentação acadêmica, 2025.

[5] I. D. Mienye, T. G. Swart, and G. Obaido, "Recurrent neural networks: A comprehensive review of architectures, variants, and applications," *MDPI-Information*, vol. 15, no. 9, 2024.

[6] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," 2015.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation, 9(8): 1735-1780*, 1997.

[8] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[9] R. Bolboacă and P. Haller, "Performance analysis of long short-term memory predictive neural networks on time series data," *MDPI-Mathematics*, vol. 11, no. 6, 2023.

[10] F. Chollet, *Deep Learning with Python*. International Conference on Machine Learning, 2 ed., 2021.

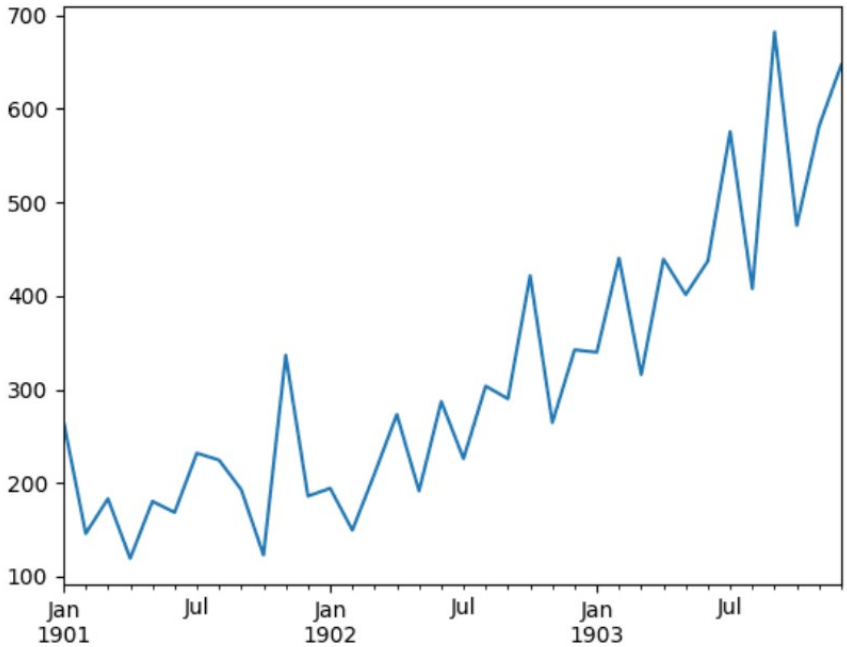
Descrição da Atividade

Baseado no código abaixo

```
from pandas import DataFrame
from pandas import Series
from pandas import concat
from pandas import read_csv
from pandas import datetime
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from math import sqrt
from matplotlib import pyplot
import numpy

# date-time parsing function for loading the dataset
def parser(x):
return datetime.strptime('190'+x, '%Y-%m')

# frame a sequence as a supervised learning problem
def timeseries_to_supervised(data, lag=1):
df = DataFrame(data)
columns = [df.shift(i) for i in range(1, lag+1)]
columns.append(df)
df = concat(columns, axis=1)
df.fillna(0, inplace=True)
return df

# create a differenced series
def difference(dataset, interval=1):
diff = list()
for i in range(interval, len(dataset)):
value = dataset[i] - dataset[i - interval]
diff.append(value)
return Series(diff)

# invert differenced value
def inverse_difference(history, yhat, interval=1):
return yhat + history[-interval]

# scale train and test data to [-1, 1]
def scale(train, test):
# fit scaler
scaler = MinMaxScaler(feature_range=(-1, 1))
scaler = scaler.fit(train)
# transform train
train = train.reshape(train.shape[0], train.shape[1])
train_scaled = scaler.transform(train)
# transform test
test = test.reshape(test.shape[0], test.shape[1])
test_scaled = scaler.transform(test)
return scaler, train_scaled, test_scaled

# inverse scaling for a forecasted value
def invert_scale(scaler, X, value):
new_row = [x for x in X] + [value]
array = numpy.array(new_row)
array = array.reshape(1, len(array))
inverted = scaler.inverse_transform(array)
return inverted[0, -1]

# fit an LSTM network to training data
def fit_lstm(train, batch_size, nb_epoch, neurons):
X, y = train[:, 0:-1], train[:, -1]
X = X.reshape(X.shape[0], 1, X.shape[1])
model = Sequential()
model.add(LSTM(neurons, batch_input_shape=(batch_size, X.shape[1], X.shape[2]), stateful=True))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
for i in range(nb_epoch):
model.fit(X, y, epochs=1, batch_size=batch_size, verbose=0, shuffle=False)
model.reset_states()
return model

# make a one-step forecast
def forecast_lstm(model, batch_size, X):
X = X.reshape(1, 1, len(X))
yhat = model.predict(X, batch_size=batch_size)
return yhat[0,0]

# load dataset
series = read_csv('caminho/nome.csv', header=0, parse_dates=[0], index_col=0, squeeze=True, date_parser=parser)

# transform data to be stationary
raw_values = series.values
diff_values = difference(raw_values, 1)
```

```
# transform data to be supervised learning
supervised = timeseries_to_supervised(diff_values, 1)
supervised_values = supervised.values

# split data into train and test-sets
train, test = supervised_values[0:-12], supervised_values[-12:]

# transform the scale of the data
scaler, train_scaled, test_scaled = scale(train, test)

# fit the model
lstm_model = fit_lstm(train_scaled, 1, 3000, 4)
# forecast the entire training dataset to build up state for forecasting
train_reshaped = train_scaled[:, 0].reshape(len(train_scaled), 1, 1)
lstm_model.predict(train_reshaped, batch_size=1)

# walk-forward validation on the test data
predictions = list()
for i in range(len(test_scaled)):
# make one-step forecast
X, y = test_scaled[i, 0:-1], test_scaled[i, -1]
yhat = forecast_lstm(lstm_model, 1, X)
# invert scaling
yhat = invert_scale(scaler, X, yhat)
# invert differencing
yhat = inverse_difference(raw_values, yhat, len(test_scaled)+1-i)
# store forecast
predictions.append(yhat)
expected = raw_values[len(train) + i + 1]
print('Month=%d, Predicted=%f, Expected=%f' % (i+1, yhat, expected))

# report performance
rmse = sqrt(mean_squared_error(raw_values[-12:], predictions))
print('Test RMSE: %.3f' % rmse)
# line plot of observed vs predicted
pyplot.plot(raw_values[-12:])
pyplot.plot(predictions)
pyplot.show()
```

Faça a avaliação dos datasets de vendas de Shampoo (vendas-shampoo.csv) que contém a série temporal abaixo

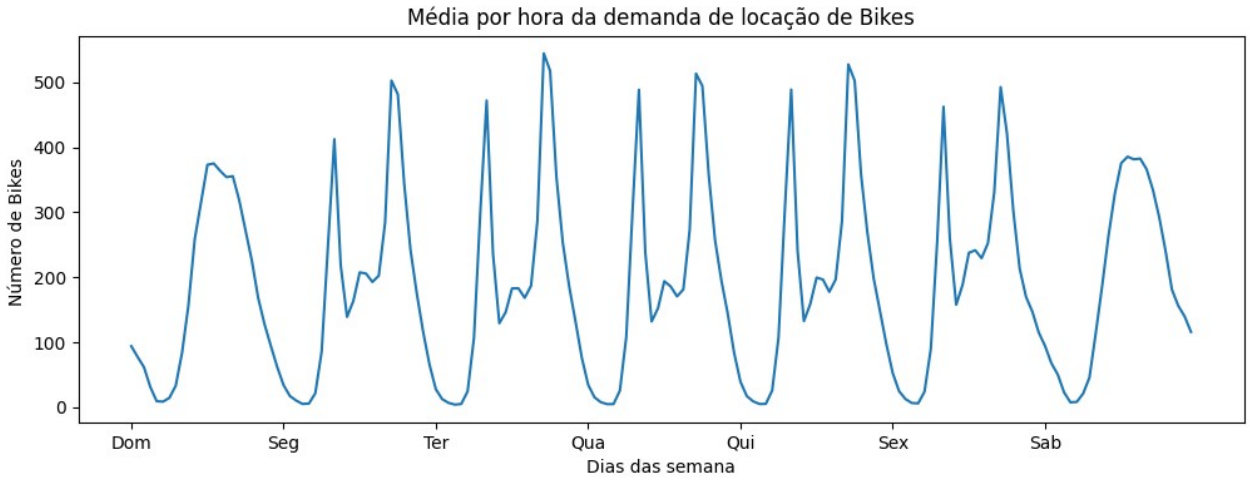


E o dataset de locação de bikes no estado de New York

from sklearn.datasets import fetch_openml
bike_sharing = fetch_openml("Bike_Sharing_Demand", version=2, as_frame=True)
df = bike_sharing.frame

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(12, 4))
average_week_demand = df.groupby(["weekday", "hour"])["count"].mean()
average_week_demand.plot(ax=ax)
_ = ax.set(
    title="Média por hora da demanda de locação de Bikes",
    xticks=[i * 24 for i in range(7)],
    xticklabels=["Dom", "Seg", "Ter", "Qua", "Qui", "Sex", "Sab"],
    xlabel="Dias das semana",
    ylabel="Número de Bikes ",
)
```

Com a seguinte série temporal



Baseado nos resultados, crie um relatório para as duas atividades e comente seus resultados e a experiência utilizando redes LTSM para capturar os padrões temporais.