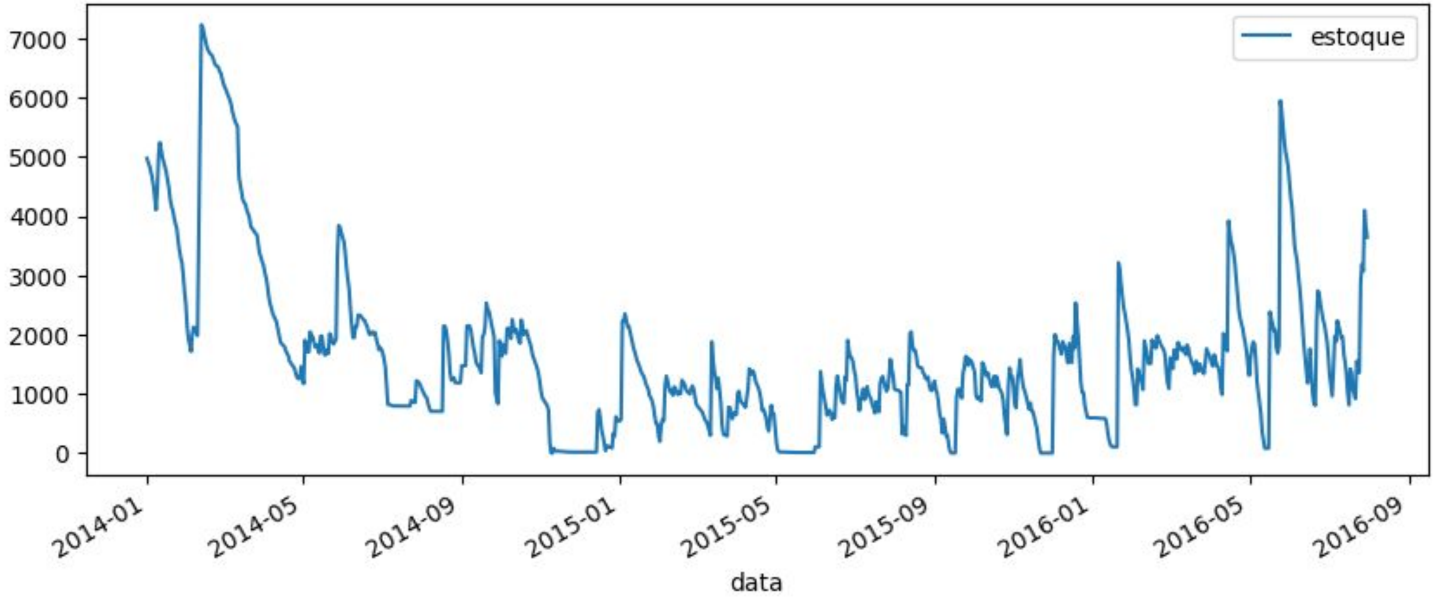
Forecasting com LSTM: prevendo estoque

Aluno: José Augusto Agripino de Oliveira
Disciplina: Tópicos especiais em Processamento Inteligente da Informação
Professor: Tiago Barros

Dataset

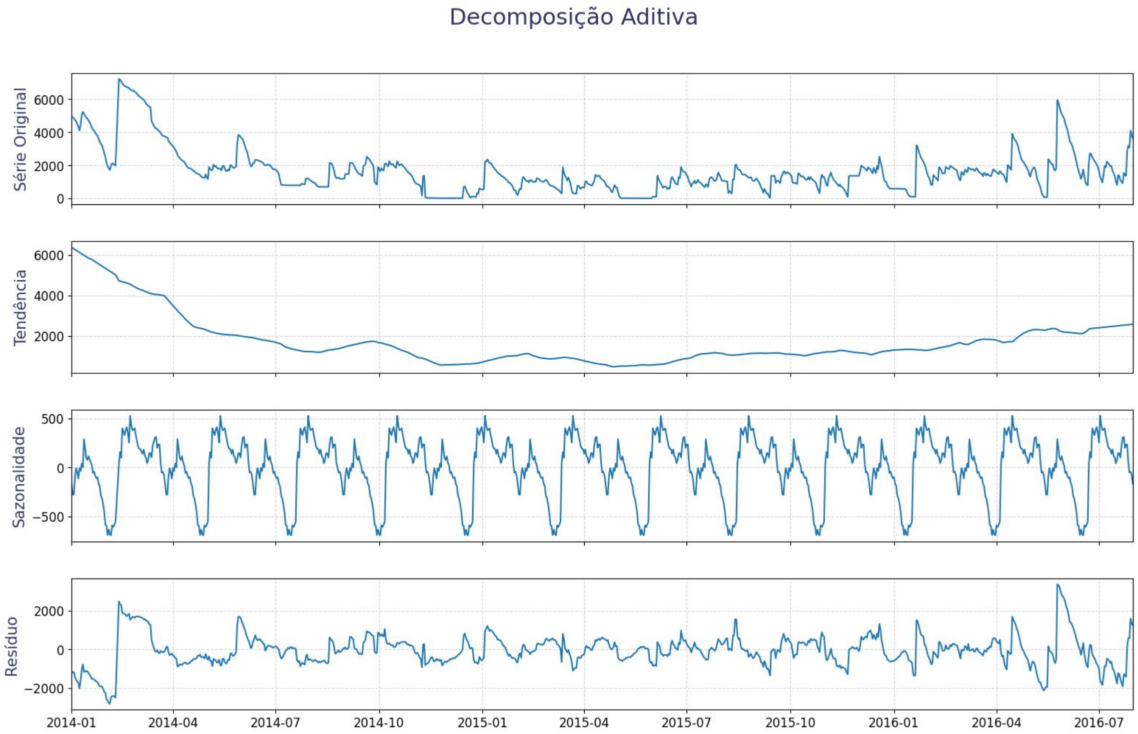
- Dataset de vendas encontrado no Kaggle;
- Há 937 amostras no dado;
- Existem as colunas: data, venda, estoque e preço;
- Foi decidido prever o estoque;
- Existem vários outliers no dado.

Dataset



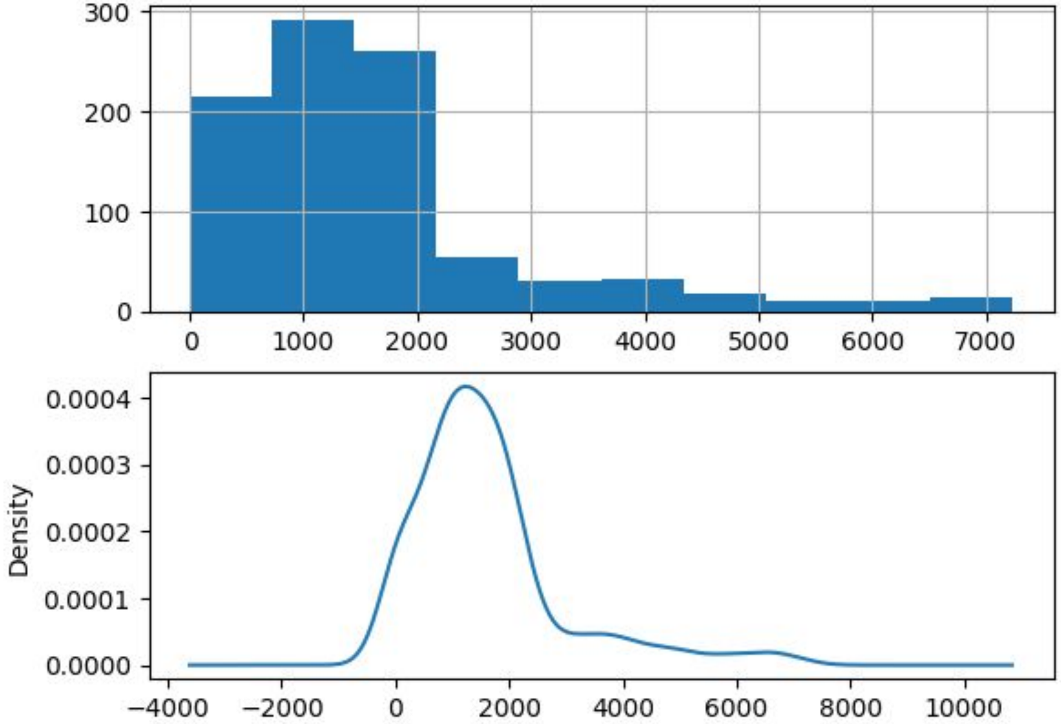
7000
6000
5000
4000
3000
2000
1000
0
estoque
2014-01
2014-05
2014-09
2015-01
2015-05
2015-09
2016-01
2016-05
2016-09
data

Dataset



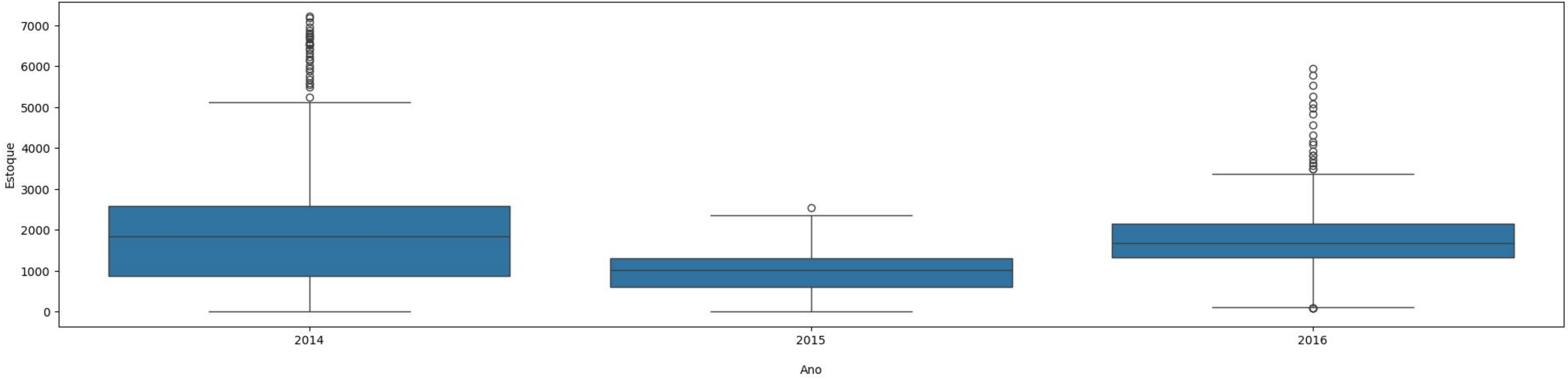
Decomposição Aditiva
Série Original
Tendência
Sazonalidade
Resíduo
6000
4000
2000
0
500
−500
−2000
2014-01
2014-04
2014-07
2014-10
2015-01
2015-04
2015-07
2015-10
2016-01
2016-04
2016-07

Dataset



300
200
100
0
0
1000
2000
3000
4000
5000
6000
7000
Density
0.0004
0.0003
0.0002
0.0001
0.0000
−4000
−2000
0
2000
4000
6000
8000
10000

Dataset


Estoque
7000
6000
5000
4000
3000
2000
1000
0
2014
2015
2016
Ano

LSTM

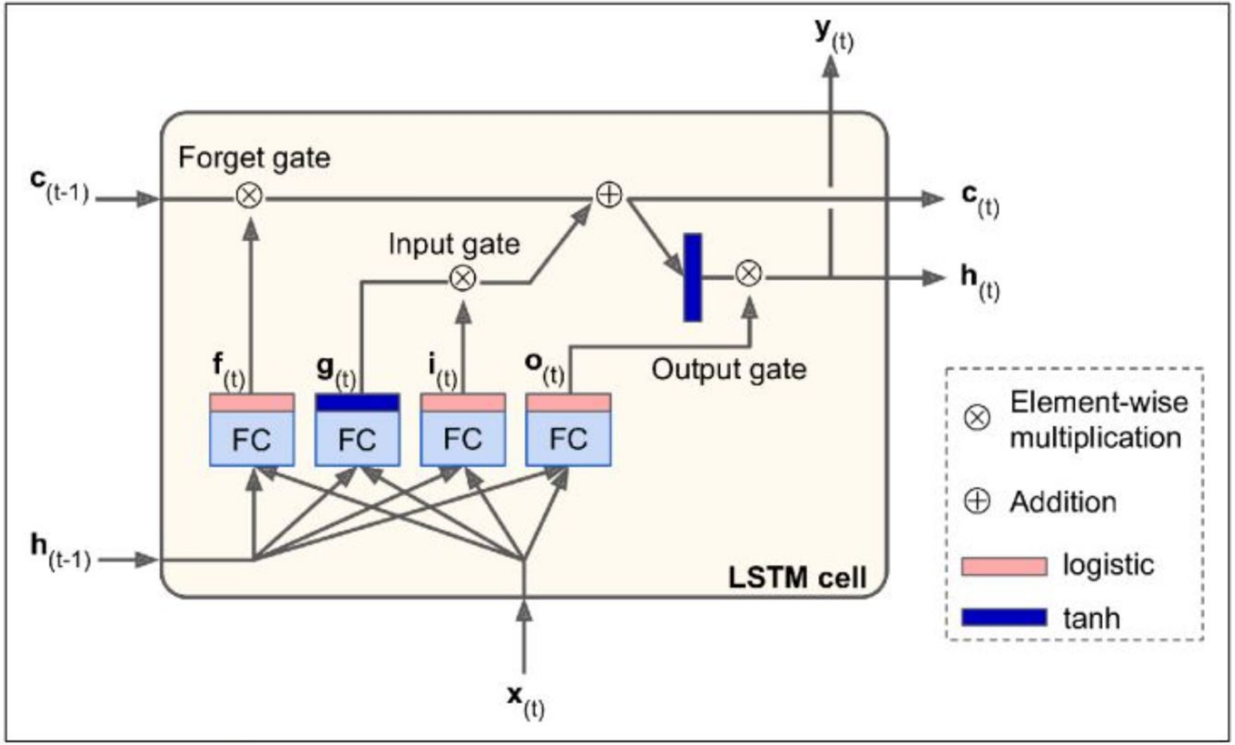


Figure 15-9. LSTM cell

Implementação

- `train_size = int(len(dados_serie) * 0.8)`
- `scaler_preditores = StandardScaler()`
- `time_steps = 24`

```
1 print(X_train.shape, y_train.shape)
```

```
(725, 24, 1) (725,)
```

```
1 print(X_test.shape, y_test.shape)
```

```
(164, 24, 1) (164,)
```

Arquitetura utilizada: Naive LSTM

```python
# Cria o modelo com LSTM simples
model_lstm = tf.keras.models.Sequential([
  tf.keras.layers.LSTM(20, input_shape=[None, 1]),
  tf.keras.layers.Dense(1)
])
```

```python
# Compila o modelo
model_lstm.compile(optimizer='adam',
                   loss=tf.keras.losses.Huber(delta=0.2),
                   metrics=['mse', 'mae'])
model_lstm.summary()
```

Arquitetura utilizada: LSTM Robusto

```
# Cria o modelo com LSTM mais robusto
model_lstm = tf.keras.models.Sequential([
  tf.keras.layers.LSTM(64, return_sequences=True, input_shape=[None, 1]),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.LSTM(32),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(16, activation='relu'),
  tf.keras.layers.Dense(1)
])
```

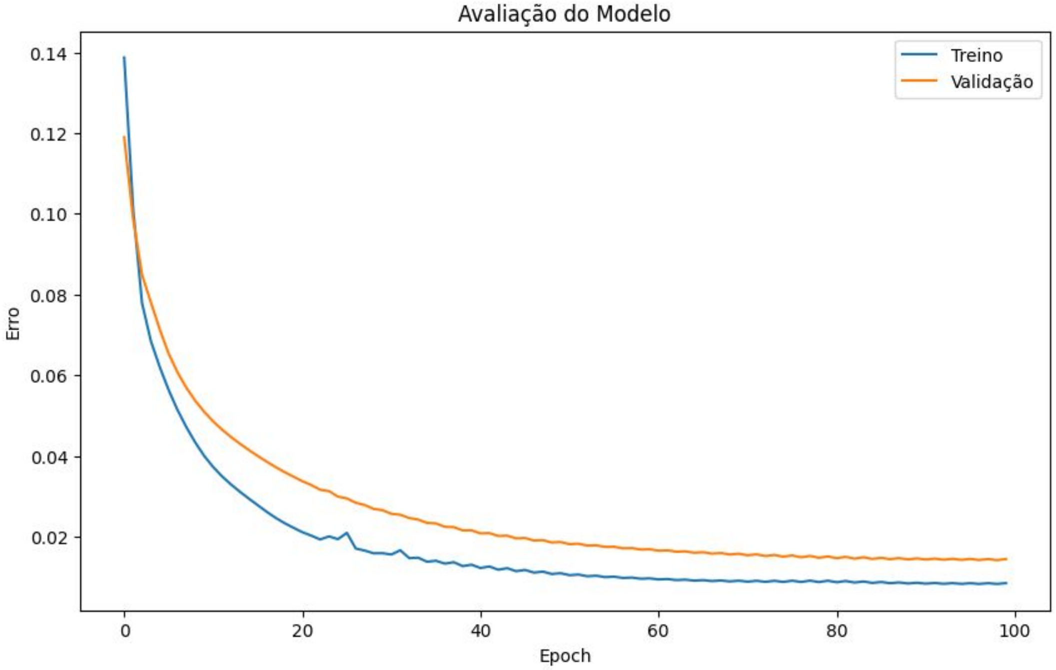
```
# Compila o modelo
optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)
model_lstm.compile(optimizer=optimizer,
                   loss=tf.keras.losses.Huber(delta=0.5),
                   metrics=['mse', 'mae'])
model_lstm.summary()
```

Naive LSTM: Treinamento

```
%%time
model_lstm_history = model_lstm.fit(X_train,
                                    y_train,
                                    epochs = 100,
                                    batch_size = 32,
                                    validation_split = 0.1,
                                    shuffle = False)
```

```
CPU times: user 26.2 s, sys: 837 ms, total: 27 s
```

Naive LSTM: Treinamento

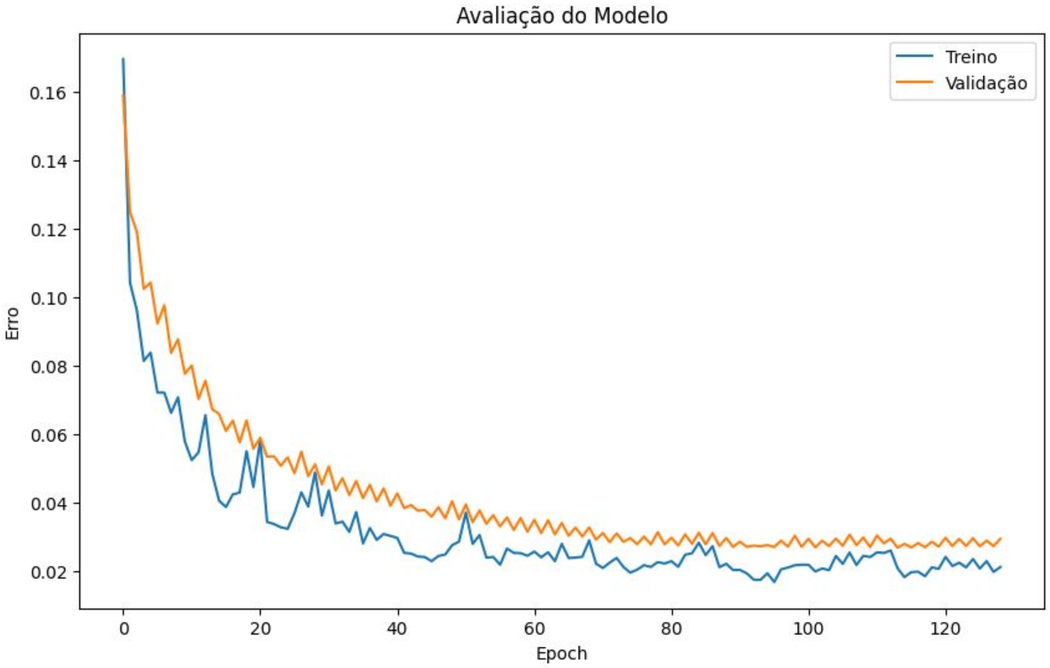


LSTM Robusto: Treinamento

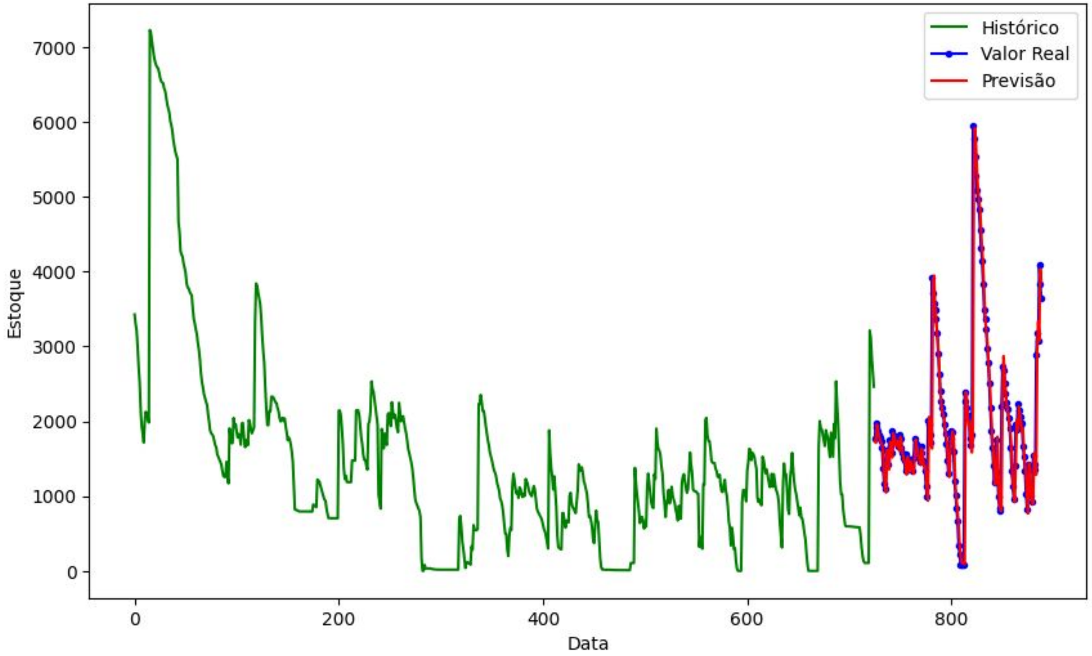
```
%%time
# Callback para early stopping
early_stop = tf.keras.callbacks.EarlyStopping(
    monitor='val_loss',
    patience=15,
    restore_best_weights=True
)

model_lstm_history = model_lstm.fit(X_train,
                                    y_train,
                                    epochs=200,
                                    batch_size=32,
                                    validation_split=0.1,
                                    shuffle=False,
                                    callbacks=[early_stop],
                                    verbose=1)
```

```
CPU times: user 1min 9s, sys: 22.7 s, total: 1min 32s
```

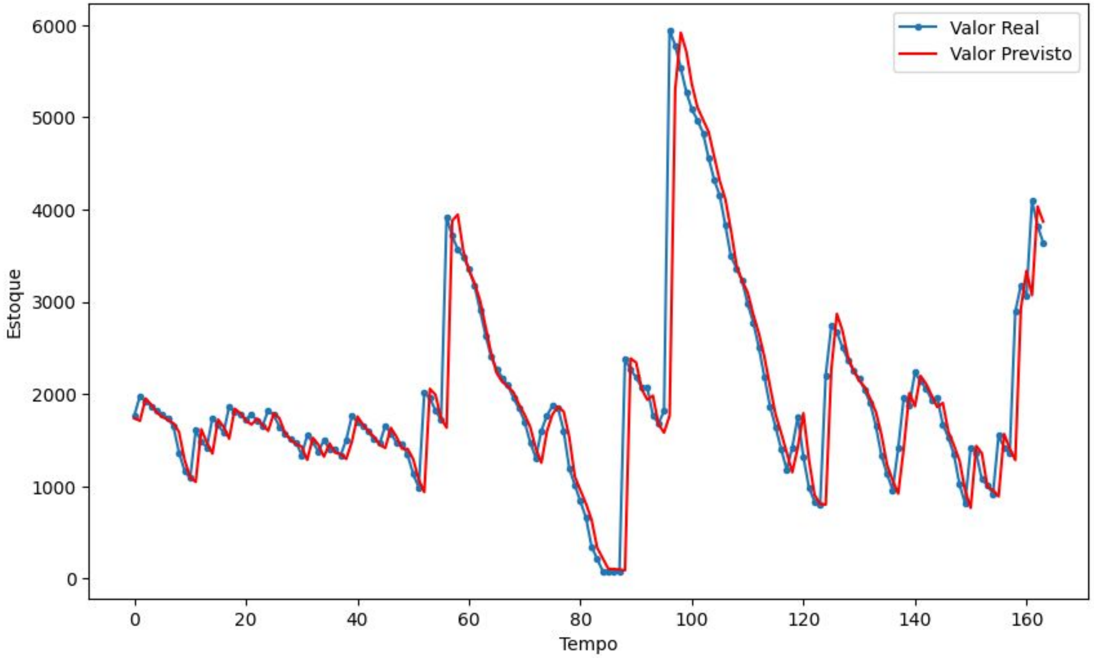
LSTM Robusto: Treinamento



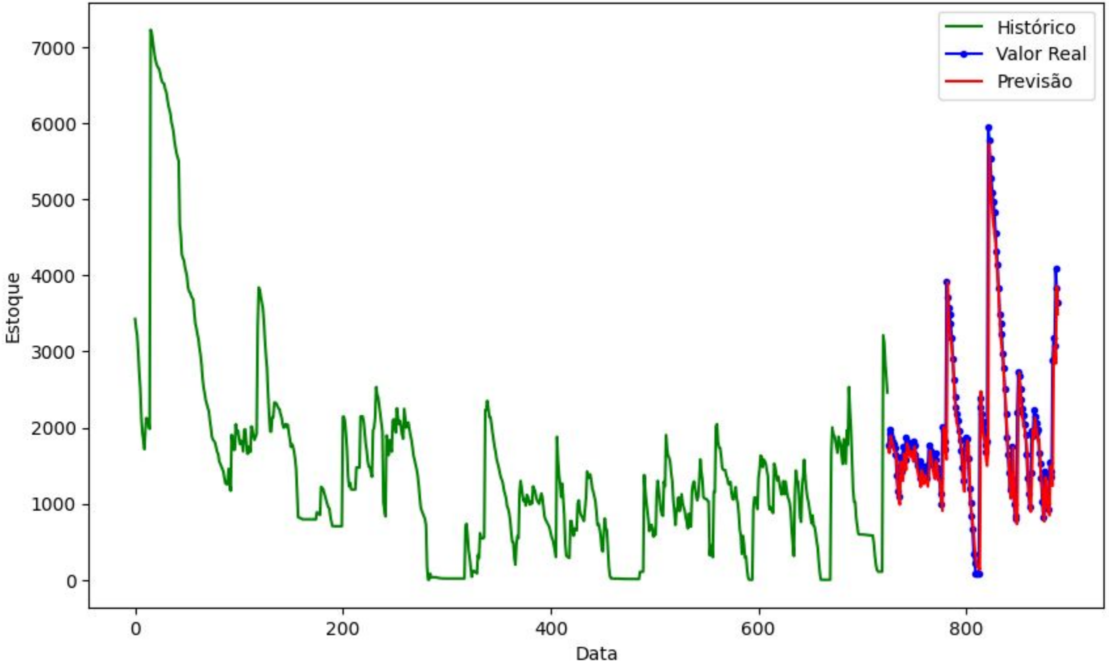
Naive LSTM: Resultados



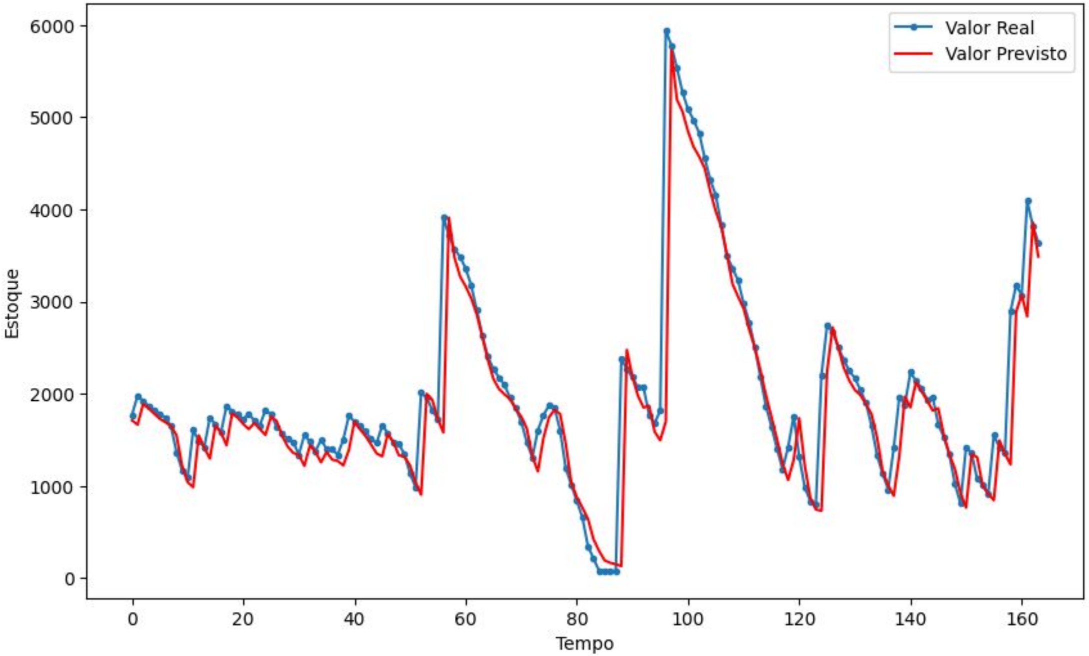
Naive LSTM: Resultados



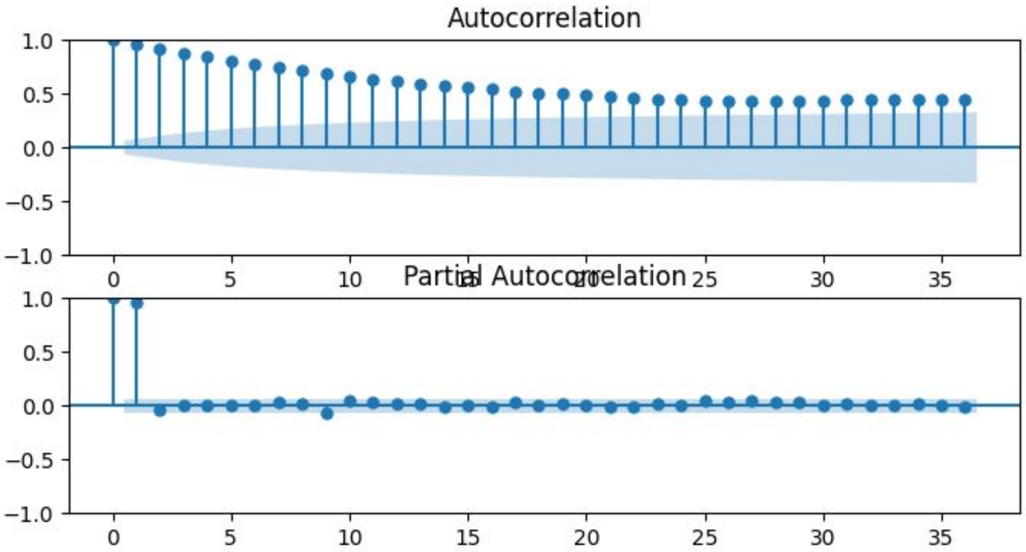
LSTM Robusto: Resultados



LSTM Robusto: Resultados

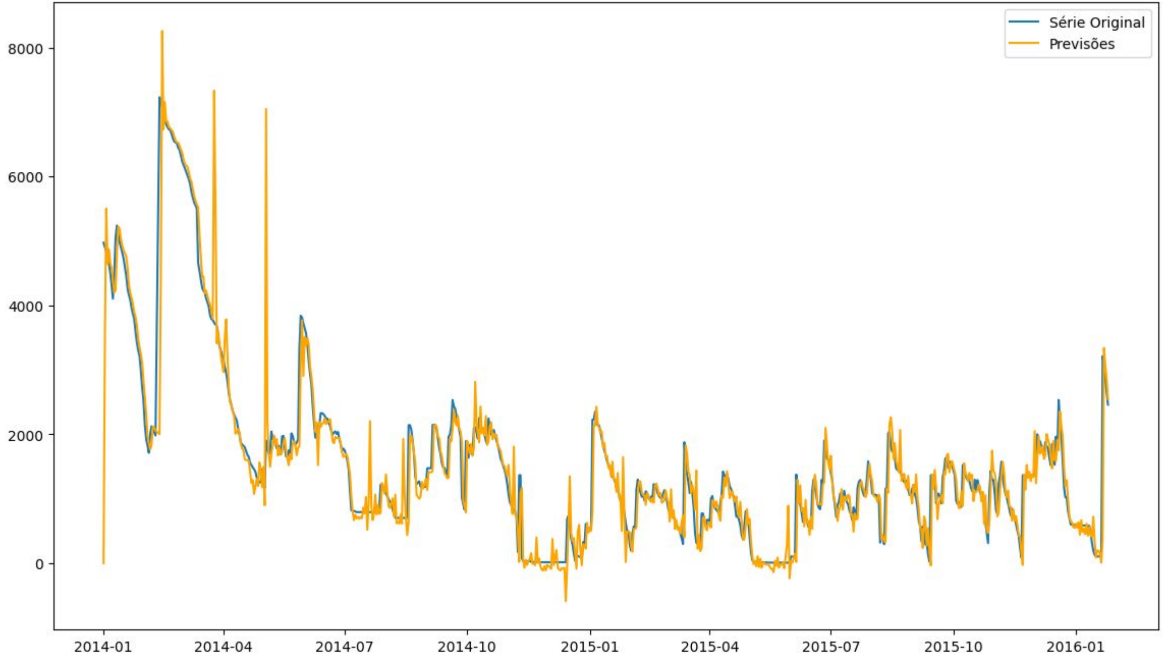


Modelo SARIMA: Escolha dos Parâmetros

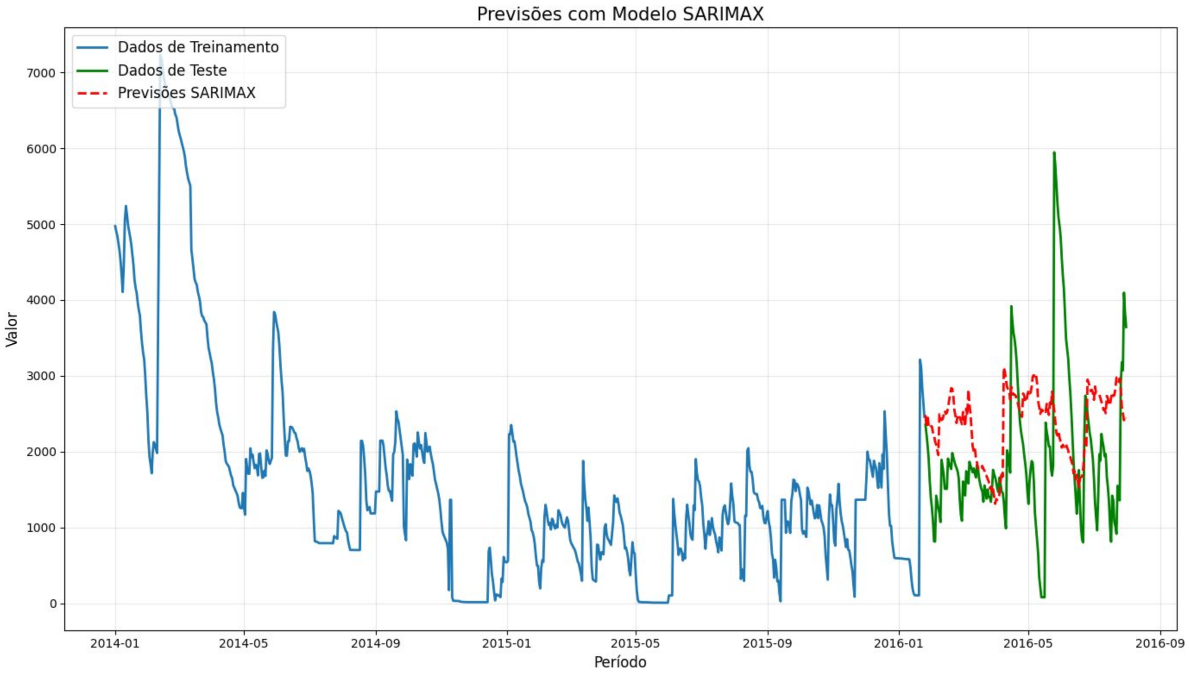



```
Total de modelos a testar: 8
Iniciando busca de parâmetros...

SARIMAX(1, 1, 2)x(0, 0, 0, 78) - AIC:10708.61 BIC:10727.06
SARIMAX(1, 1, 2)x(0, 0, 1, 78) - AIC:9318.82 BIC:9341.33
SARIMAX(1, 1, 2)x(0, 1, 0, 78) - AIC:9830.41 BIC:9848.42
SARIMAX(1, 1, 2)x(0, 1, 1, 78) - AIC:8478.89 BIC:8500.78
SARIMAX(1, 1, 2)x(1, 0, 0, 78) - AIC:9337.60 BIC:9360.13
SARIMAX(1, 1, 2)x(1, 0, 1, 78) - AIC:9311.38 BIC:9338.39
SARIMAX(1, 1, 2)x(1, 1, 0, 78) - AIC:8522.71 BIC:8544.62
SARIMAX(1, 1, 2)x(1, 1, 1, 78) - AIC:8444.68 BIC:8470.95
```

Modelo SARIMA: Previsão em treinamento



Modelo SARIMA: Previsão em teste



Comparação dos Modelos

Métrica	Naive LSTM	LSTM Robusto	SARIMA
MAE	205.89164445458388	207.94989683569932	957.3295072099455
MSE	247,468.42987973997	254,300.8778883564	1,446,589.3466982
RMSE	497.4619883767402	504.28253775870166	1202.7424274125362
EVS	0.7993868948489111	0.7991748201716878	-0.18468876389992417
R^2	0.7902579029779476	0.7844670553380375	-0.3478089876325352

Conclusão

- Os resultados para o Naive LSTM e para o LSTM Robusto foram muito similares;
- Ambos os modelos LSTM testados tendem a repetir a amostra anterior;
- O modelo SARIMA está acrescentando ruída à previsão, o que torna sua previsão pouco confiável.

Referências

- https://www.kaggle.com/datasets/tevecsystems/retail-sales-forecasting/data
- BARROS, Tiago. **Aula 6 - Previsão de séries temporais com RNNs**. Natal: Ufrn, 2025. Color.