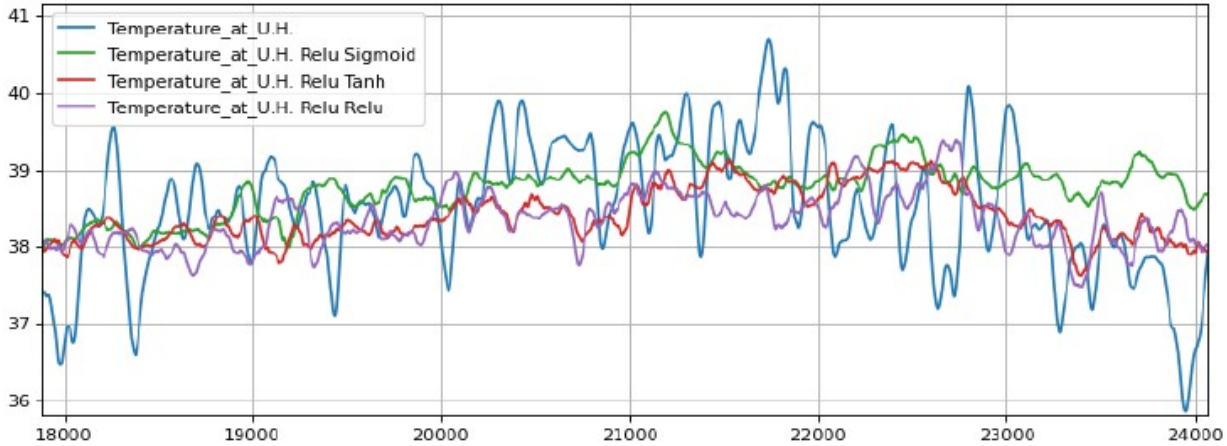
"The best and most beautiful things in this world cannot be seen or even heard but must be felt with the heart."– Helen Keller

Séries temporais



Mestrado em Engenharia Informática

2º ano, 1º semestre

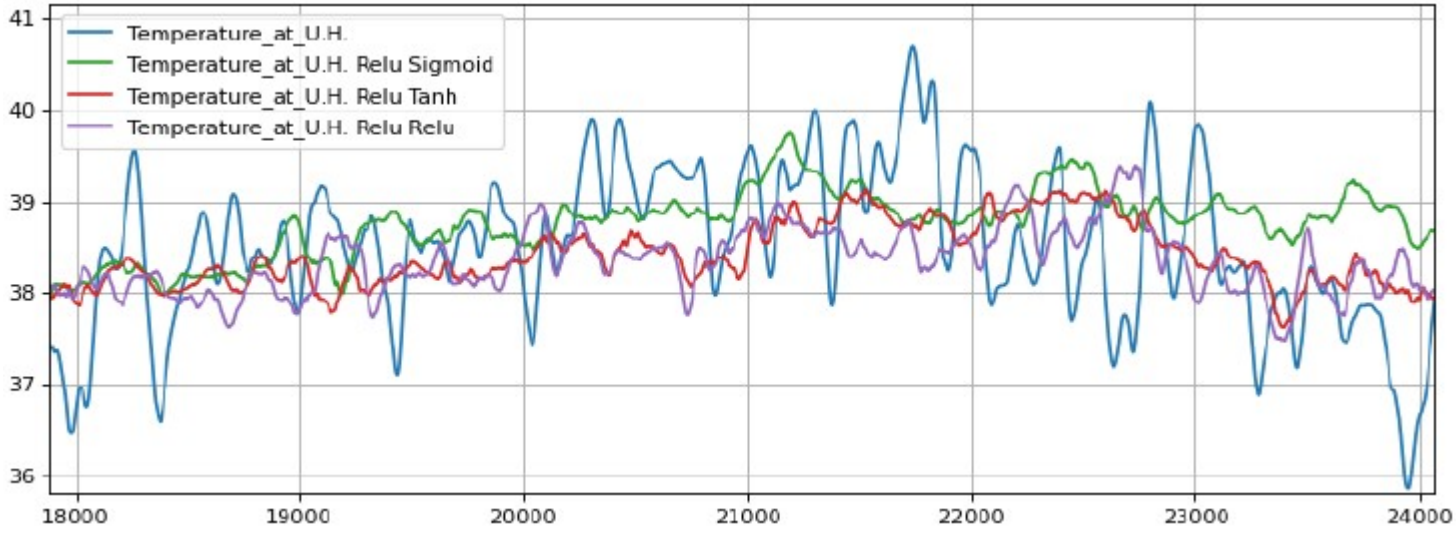
Agenda

- Séries temporais
- Predição com ARIMA
- Predição com LSTM e GRU

Séries temporais

- Muitas variáveis são registadas como sequências de valores ao longo do tempo, com um determinado período
- Exemplos: leituras de sensores como temperatura atmosférica, vibração de um equipamento, velocidade de rotação de um motor, etc.

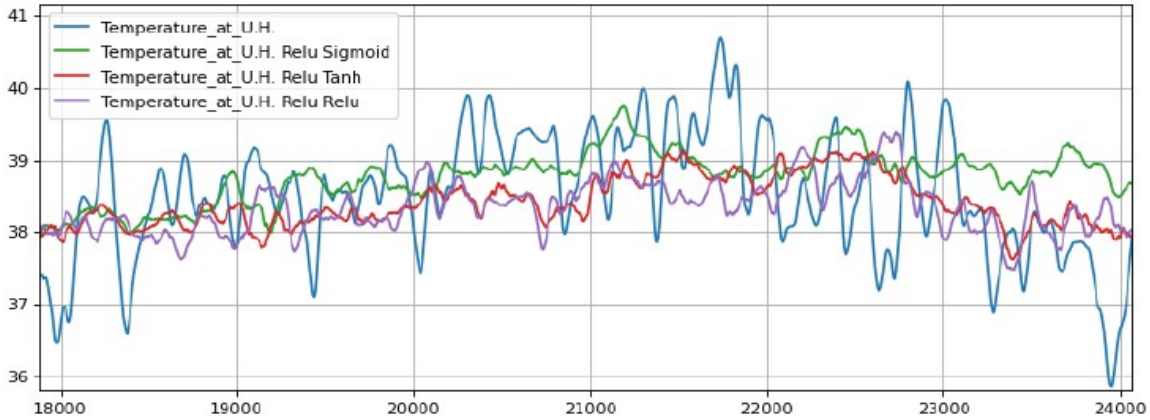
Exemplo de
série temporal de temperatura e predição usando LSTM com diferentes funções de transferência.



Mateus, B. C., Mendes, M., Farinha, J. T., & Cardoso, A. M. (2021). Anticipating future behavior of an industrial press using LSTM networks. Applied Sciences, 11(13), 6101.

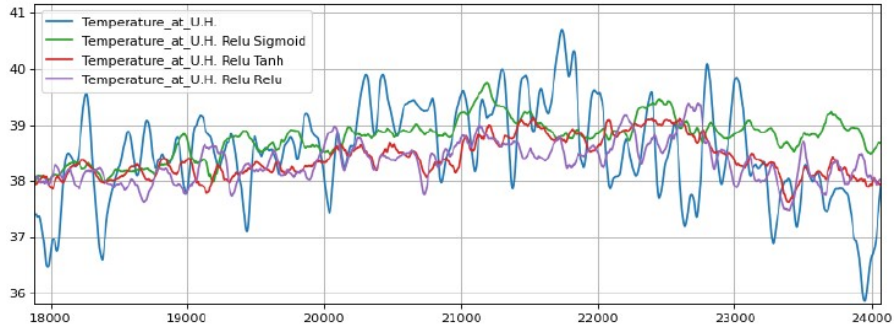
Séries temporais

- A análise de séries temporais é importante para
  - Monitorizar processos, equipamentos, fenómenos como o tempo (temperatura, humidade, etc.)
  - Detetar padrões específicos ou estados (e.g., estado de falha de um equipamento)
  - Antecipar o futuro (prever o tempo, determinar provável falha de equipamento antes de ocorrer, etc.)



Séries temporais

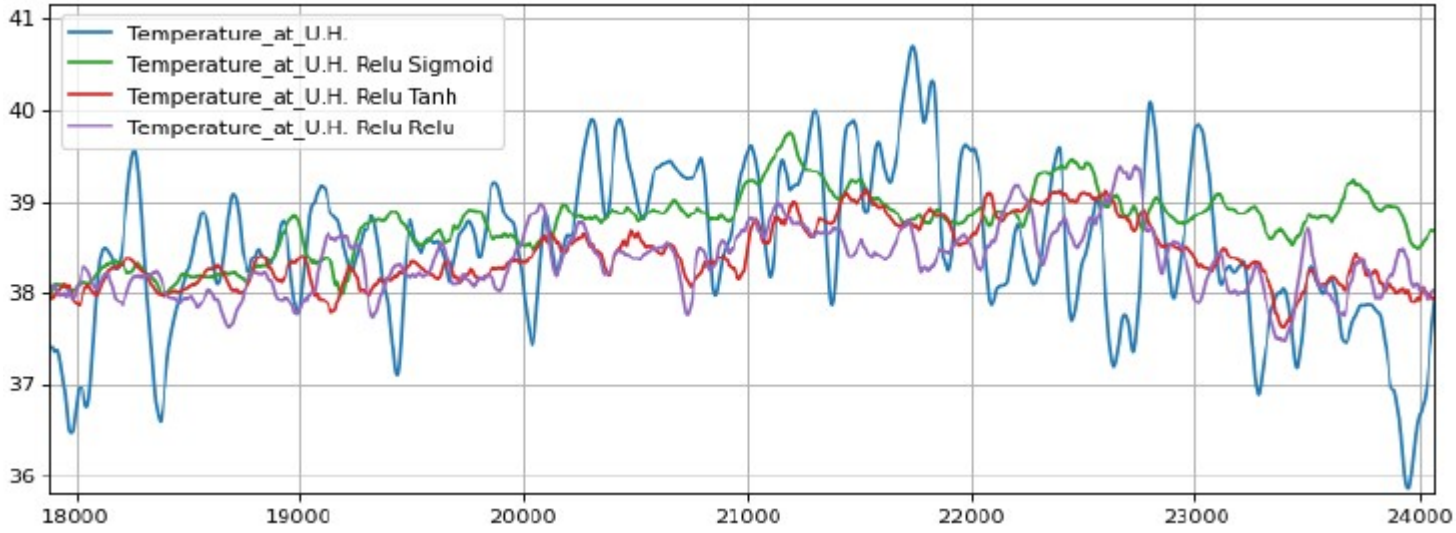
- Há diferentes métodos de prever a probabilidade de um evento ocorrer no futuro
  - Cadeias de Markov Escondidas (HMM)
  - LSTM
  - Arima, sarima, ...
  - ...



- Além da previsão, é necessário normalmente classificar o resultado da previsão
  - Clustering e SVM ou ANN são normalmente boas escolhas

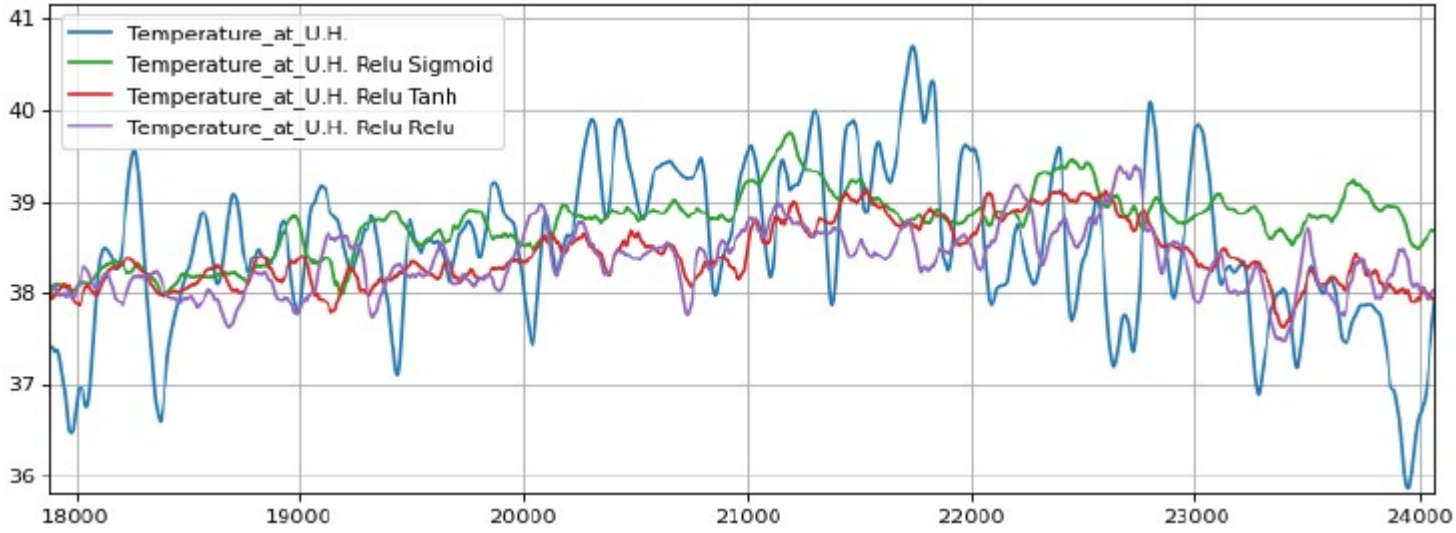
Séries temporais

- Aplicações típicas
  - Previsão da probabilidade de falha de equipamentos
  - Previsão do tempo
  - Previsão dos níveis de poluição numa cidade
  - ...



Métodos típicos de previsão

- ARIMA (e família)
  - AutoRegressive Integrated Moving Average
  - Método muito popular, baseado nas propriedades matemáticas do próprio sinal
  - O valor da amostra seguinte é estimado a partir das caraterísticas das amostras anteriores
- LSTM (e família)
  - Redes neuronais
  - Baseados em aprendizagem computacional
- Random Forest
- ...



Mas antes de criar modelos...

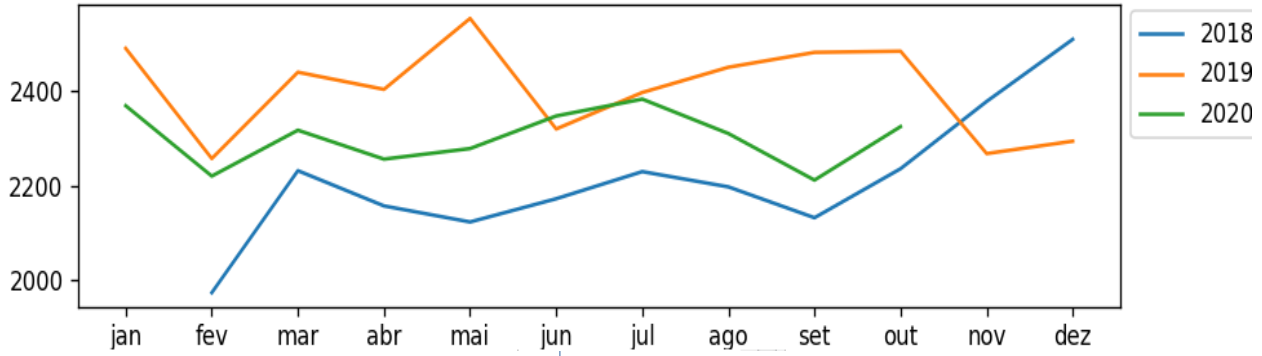
- Seguir a metodologia CRISP:
  - compreender o problema
  - explorar os dados
  - preparar os dados
  - ...

Antes de criar modelos, compreender os dados

- Fazer gráficos diversos, compreender como os dados se comportam
  - Gráficos de dados agregados de diferentes formas, aplicar filtros de média móvel, mediana
  - Gráficos year over year, week over week, etc.

```
df['year'] = [d.year for d in df.index]
df['month'] = [d.strftime('%b') for d in df.index]
years = df['year'].unique()

colname = df.columns[1]
dfp = pd.pivot_table(data=df, index='month', columns='year', values=colname, aggfunc='sum')

from calendar import month_abbr
dfp = dfp.loc[month_abbr[1:]] # sort dfp index by month

print(dfp)

ax = dfp.plot(ylabel='Aggregated Sum', figsize=(6, 4))
ax.set_xticks(range(12))  # set ticks for all months
ax.set_xticklabels(dfp.index)  # label all the ticks
ax.legend(bbox_to_anchor=(1, 1.02), loc='upper left')
```

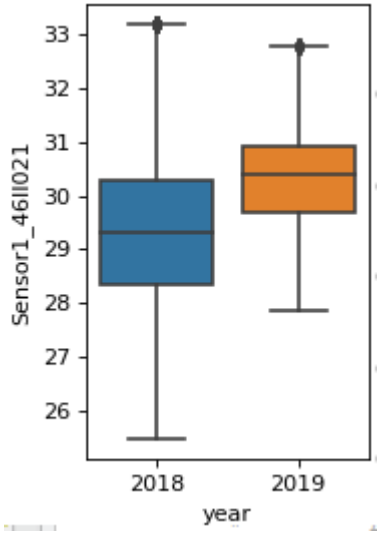
year	2018	2019	2020
month			
jan	NaN	2488.850526	2368.050851
fev	1974.273397	2256.730350	2219.825988
mar	2231.240339	2438.655644	2316.466860
abr	2157.012630	2402.479338	2255.568965
mai	2123.167491	2551.925166	2277.781333



Antes de criar modelos, compreender os dados

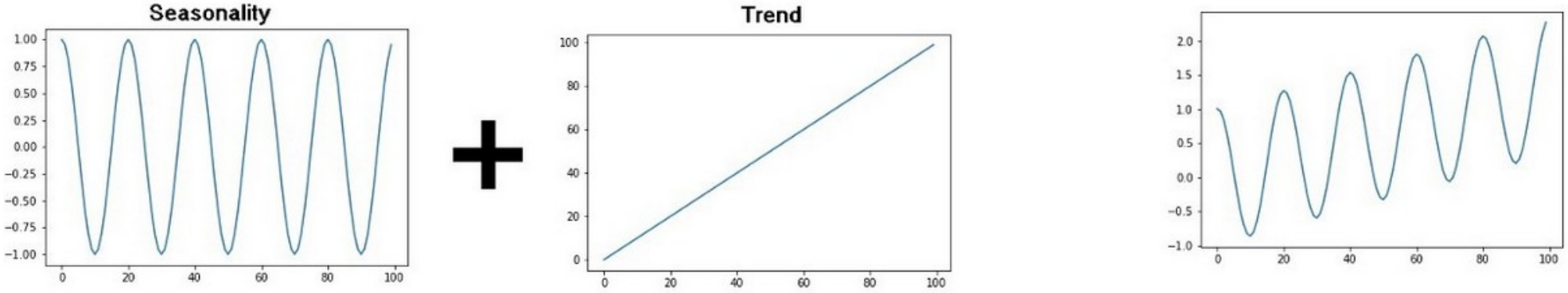
- Boxplots mostram os quartis e extremos

```
df['year'] = [d.year for d in df.Date]
df['month'] = [d.strftime('%b') for d in df.Date]
years = df['year'].unique()

colindex = 0
colname = df.columns[colindex]

# Draw Plot
fig, axes = plt.subplots(1, 2, figsize=(20,7), dpi= 80)
#sns.boxplot(x='year', y='value', data=df, ax=axes[0])
sns.boxplot(x='year', y=colname,data = df, ax=axes[0])
```



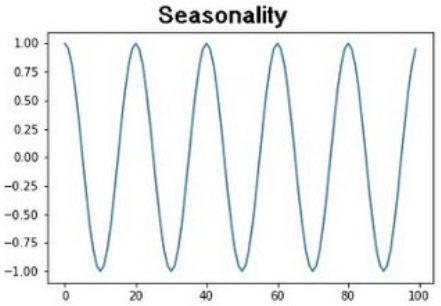
Sasonalidade (Aditiva)

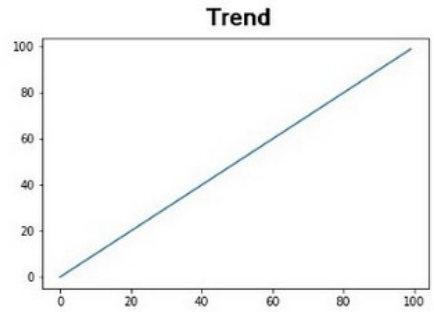
- Decompor os dados em nível base + tendência + componente sasonal + resíduo
  - Base level: constante
  - Trend: parte que sobe ou desce de forma contínua
  - Seasonality: padrão que se repete com determinada periodicidade (ano, mês, dia, hora, etc.
  - Sasonalidade aditiva pode afetar a tendência



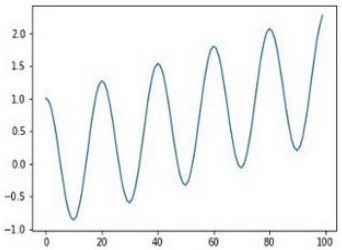
Sasonalidade multiplicativa

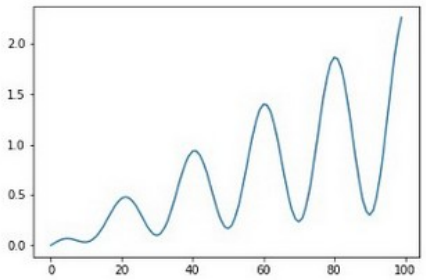
- Multiplicativa: Base level *x* trend *x* seasonal component *x* Residue
- Afeta a amplitude
- A observação dos dados pode revelar logo a decomposição mais adequada
- Conhecendo o problema pode-se ter pistas dos padrões a procurar. Ex.: fábrica com turno de 8 h é provável que tenha padrões com esse período





Additive



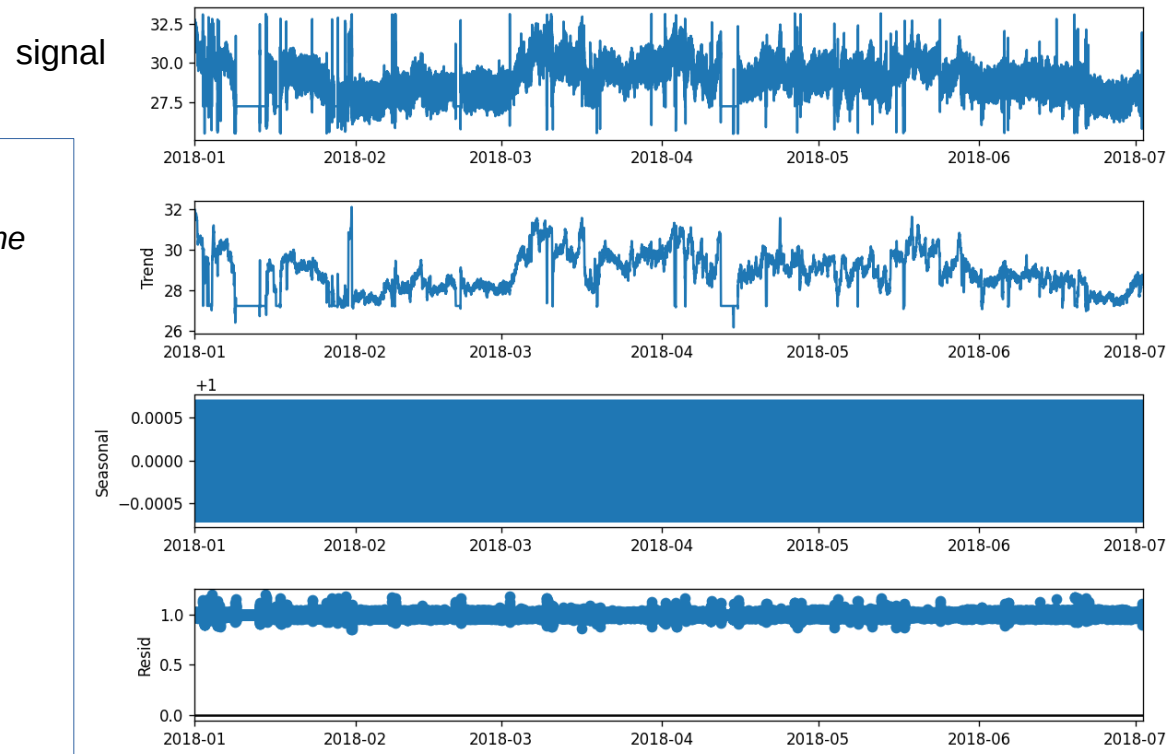


https://towardsdatascience.com/finding-seasonal-trends-in-time-series-data-with-python-ce10c37aa861

Decomposição sasonal

- Em python feita com seasonal_decompose

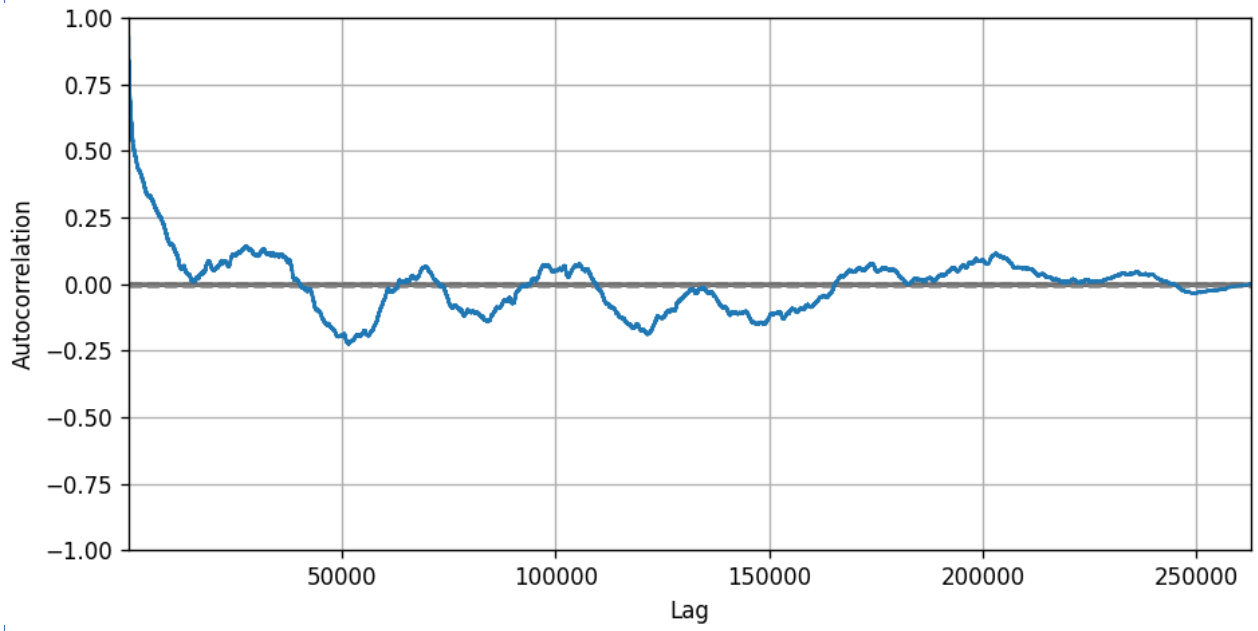
```
from statsmodels.tsa.seasonal import seasonal_decompose

df1 = df18.copy()[:262800] # Get part of the lines in the dataframe

df1.set_index('Date', inplace=True)

df2 = df1[[colname]].copy() # Select relevant column

decompose_result_mult = seasonal_decompose(df2,
                                           model="multiplicative", period=60)

trend = decompose_result_mult.trend
seasonal = decompose_result_mult.seasonal
residual = decompose_result_mult.resid

decompose_result_mult.plot();
```



O gráfico de autocorrelação também dá pistas

- A forma como a autocorrelação evolui em função das *lags* revela provável período do padrão sasonal

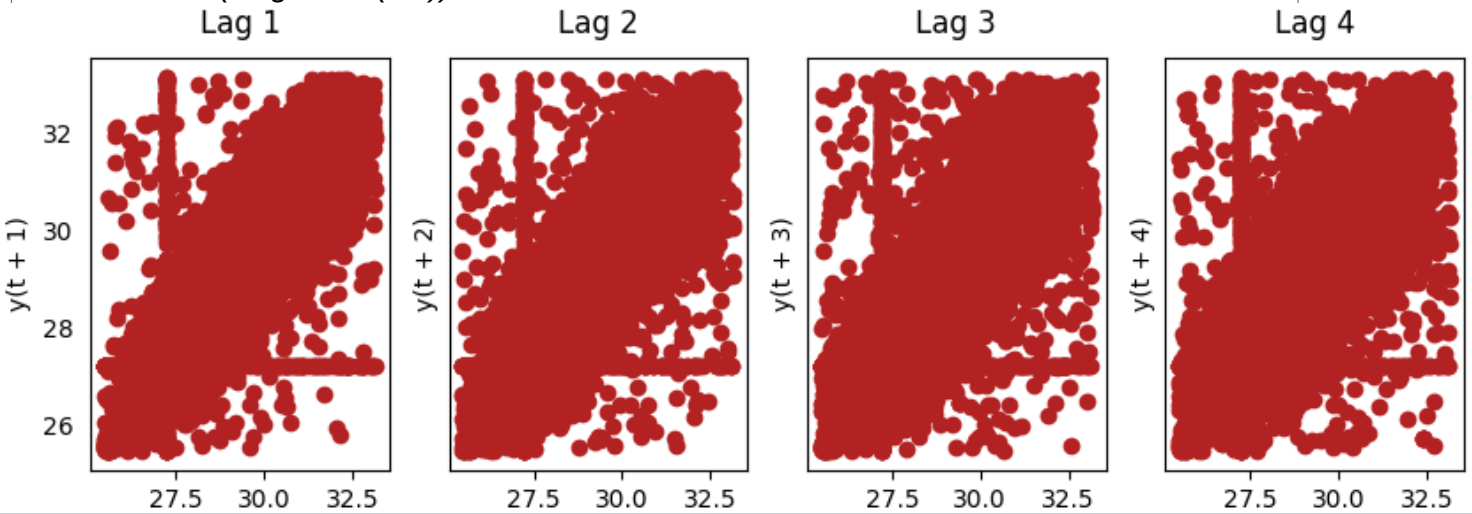
```
from pandas.plotting import autocorrelation_plot

plt.rcParams.update({'figure.figsize':(9,5),
'figure.dpi':120})

autocorrelation_plot(df2)
```



Lag plots – outra forma de verificar auto-correlações

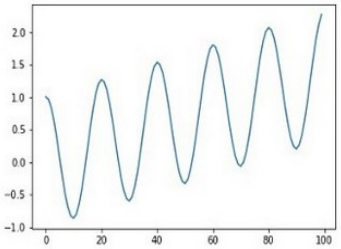
- Gráfico de amostra *n* contra amostra *n-k* : espalhamento e outros padrões revelam informação do sinal

```
from pandas.plotting import lag_plot
plt.rcParams.update({'ytick.left' : False, 'axes.titlepad':10})


ig, axes = plt.subplots(1, 4, figsize=(10,3), sharex=True, sharey=True, dpi=100)
for i, ax in enumerate(axes.flatten()[:4]):
    lag_plot(df2, lag=i+1, ax=ax, c='firebrick')
    ax.set_title('Lag ' + str(i+1))
```



Estacionaridade

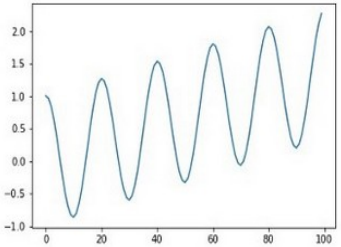
- Uma série é estacionária se a média, variância e autocorrelação não dependem do tempo
- Séries não estacionárias são difíceis de prever
- Para fazer uma série estacionária
  - Diferenciar uma ou mais vezes
  - Usar o logaritmo
  - Usar a raiz
  - Usar duas ou três técnicas acima



Mean increases, so this series is not stationary

Estacionaridade

- Testar estacionaridade
  - A partir do gráfico ou técnicas anteriores
  - Dividir a série em partes e calcular média, variância e auto-correlação
  - Usar os testes
    - Augmented Dickey Fuller – ADF
    - Kwiatkowski-Phillips-Schmidt-Shin – KPSS



Mean increases, so this series is not stationary

Estacionaridade

- Teste ADF:
  - Testa se existe uma raiz unitária
    - O número de raizes unitárias equivale ao número de vezes que a série precisa de ser diferenciada para se tornar estacionária
  - Quando p-value < 0.05, aceita-se que a série será estacionária
    - quando é maior assume-se que não é estacionária e precisa de ser diferenciada para o ser

```
from statsmodels.tsa.stattools import adfuller, kpss

result = adfuller(df2, autolag='AIC') # choose number of lags to minimize criterion

print(f'ADF Statistic test: {result[0]}')
print(f'p-value: {result[1]}')
for key, value in result[4].items(): # Critical values at 1, 5, 10 %
    print('Critial Values:')
    print(f'   {key}, {value}')
```

```
ADF Statistic: -9.789071008187635
p-value: 6.42562008514058e-17
Critial Values:
   1%, -3.430374891664653
Critial Values:
   5%, -2.8615510018008896
Critial Values:
   10%, -2.5667758558605485
```

Stationary time series

Preparar os dados

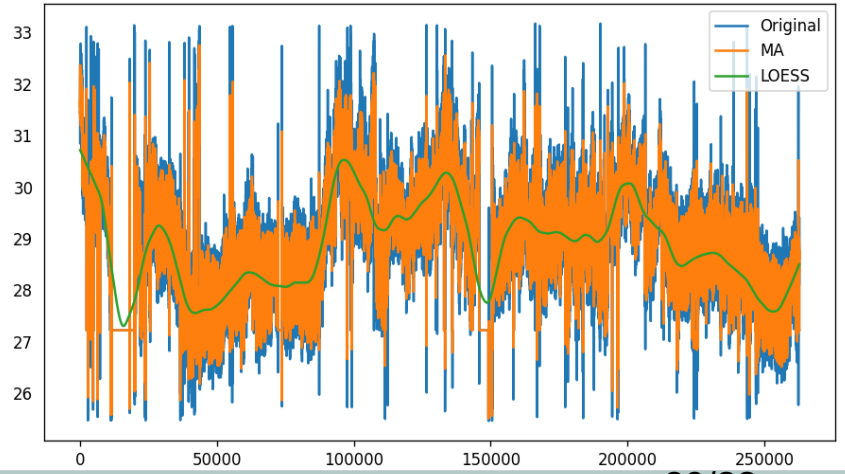
- Tratar, se necessário, dados em falta ou discrepantes
- Métodos comuns de imputação de dados em falta
  - Evitar substituir pela média, especialmente em séries não estacionárias, mas pode ser feito
  - Forward fill, backward fill são estratégias também comuns
  - Interpolação linear, quadráticas, etc.
  - Média de *n* vizinhos
  - ...

```
df_ffill = df.ffill() # forward fill

df_bfill = df.bfill() # backward fill

# See for example https://www.machinelearningplus.com/time-series/time-series-analysis-python/ for other methods
```

Alisamento (*Smoothing*)

- Melhora clareza na visualização de dados
- Reduz efeitos do ruído e dados discrepantes
- Muitas vezes facilita o processo de aprendizagem computacional
- Métodos comuns:
  - Média móvel de janela de *n* amostras; *Localized Regression* (LOESS) (regressão linear de janela de *n amostras*

```
from statsmodels.nonparametric.smoothers_lowess import lowess
df_ma = df2.rolling(3, center=True, closed='both').mean()
# 2. Loess Smoothing 5%
df_loess_5 = pd.DataFrame(lowess(df2[colname], np.arange(len(df2)),
frac=0.05)[:, 1])

plt.plot(df3,label='Original')
plt.plot(df_ma,label='MA')
plt.plot(df_loess_5,label='LOESS')
plt.legend()
```



ARIMA

- Autoregressive Integrated Moving Average
  - Adequada para séries não sasonais e não aleatórias
- Y(t) = Constante + combinação linear de *p lags* + Combinação linear de erros de predição de até *q lags*

$$Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + .. + \beta_p Y_{t-p} \epsilon_t + \phi_1 \epsilon_{t-1} + \phi_2 \epsilon_{t-2} + .. + \phi_q \epsilon_{t-q}$$

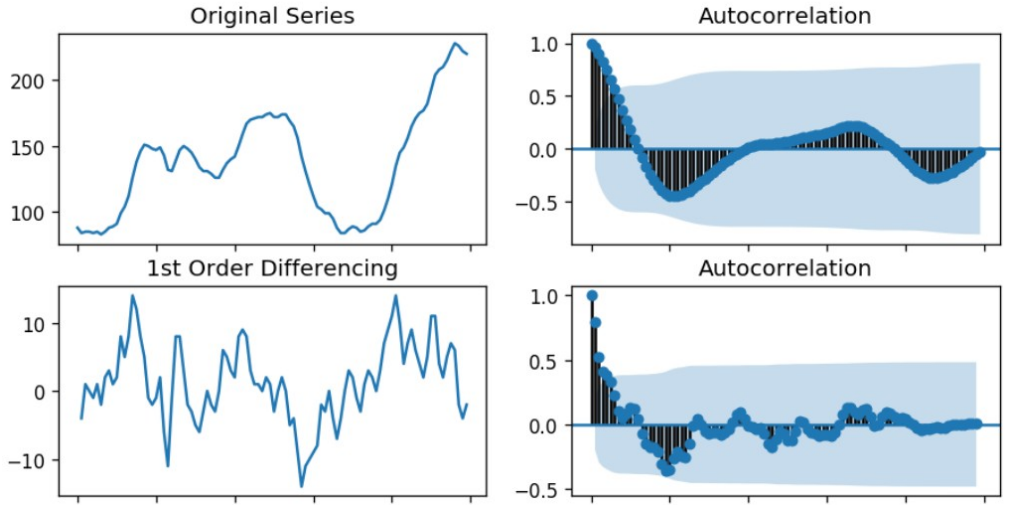
ARIMA

- Parâmetros
  - p → ordem do termo autoregressivo; nº de *lags* a usar como preditores
  - q → ordem da média móvel; nº de *lags* da média móvel
  - d → nº de vezes que é necessário diferenciar para tornar a série estacionária (0 para séries estacionárias)
- SARIMA - inclui ainda componente sasonal

ARIMA – Determinação dos parâmetros

- d → n.º de vezes a diferenciar até atingir estacionaridade
  - Usar 0 para séries estacionárias
  - Usar n.º em que a auto-correlação se torna negativa em 3 a 9 *lags*
  - Diferenciar até obter p<0.05 no teste ADF

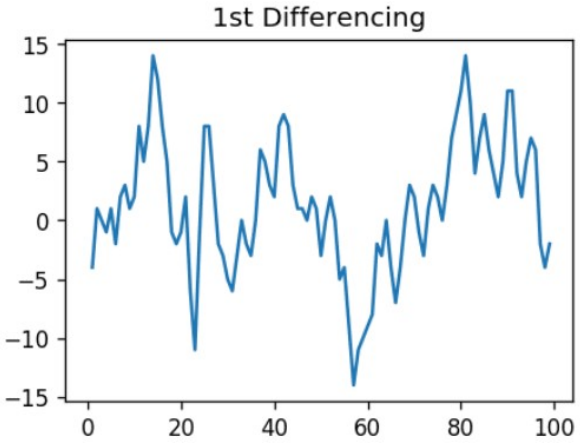
```
plt.rcParams.update({'figure.figsize':(9,7), 'figure.dpi':120})


# Original Series
fig, axes = plt.subplots(3, 2, sharex=True)
axes[0, 0].plot(df3); axes[0, 0].set_title('Original Series')
plot_acf(df3, ax=axes[0, 1])

# 1st Differencing
axes[1, 0].plot(df3.diff()); axes[1, 0].set_title('1st Order Differencing')
plot_acf(df3.diff().dropna(), ax=axes[1, 1])
```

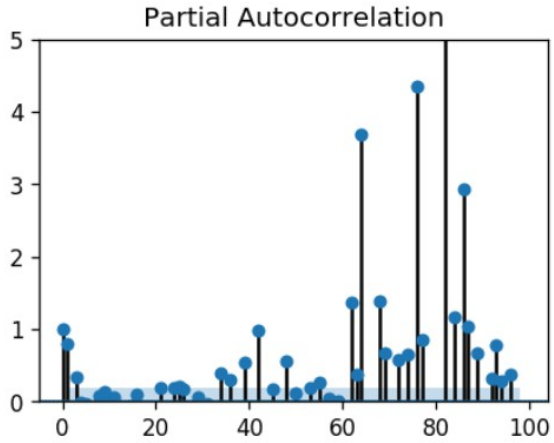


ARIMA – Determinação dos parâmetros

- p → ordem do termo autoregressivo; nº de lags a usar como preditores
  - Verificar gráfico de auto-correlação parcial
  - Escolher 1, 2, …, de forma a incluir toros os termos com autocorrelação > 0.05. Na dúvida começar com o modelo mais simples (menor p)
  - Usar p maior se a série está pouco diferenciada

```
plt.rcParams.update({'figure.figsize':(9,3), 'figure.dpi':120})

fig, axes = plt.subplots(1, 2, sharex=True)
axes[0].plot(df3.diff()); axes[0].set_title('1st Differencing')
axes[1].set(ylim=(0,5))
plot_pacf(df3.diff().dropna(), ax=axes[1])

plt.show()
```

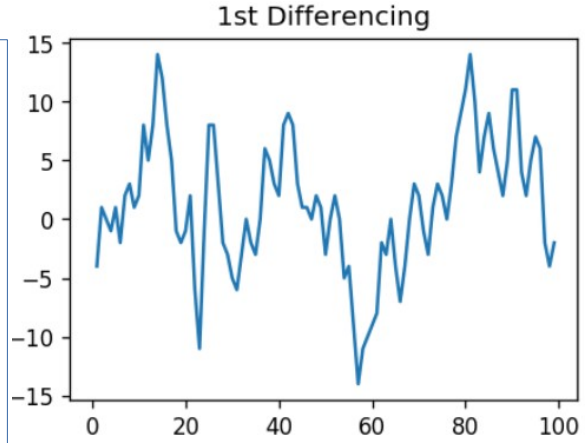


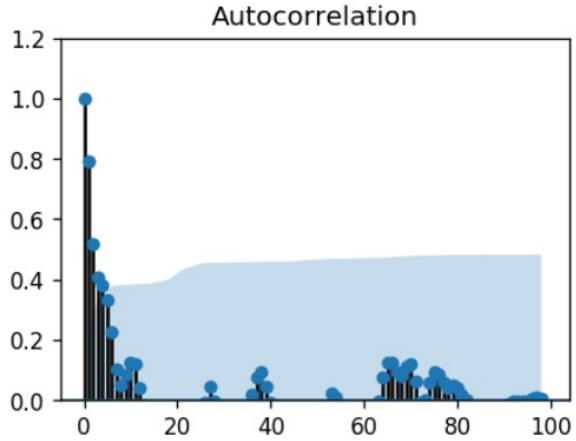


ARIMA – Determinação dos parâmetros

- q → Ordem da média móvel
  - Verificar a partir do gráfico de autocorrelação
  - Escolher 1, 2, …, de forma a incluir termos com autocorrelação acima do nível de significância 0.05
  - Na dúvida começar com modelo mais simples (q mais baixo)
  - Usar maior q se a série está pouco diferenciada

```
fig, axes = plt.subplots(1, 2, sharex=True)
axes[0].plot(df3.diff()); axes[0].set_title('1st Differencing')
axes[1].set(ylim=(0,1.2))
plot_acf(df3.diff().dropna(), ax=axes[1])

plt.show()
```

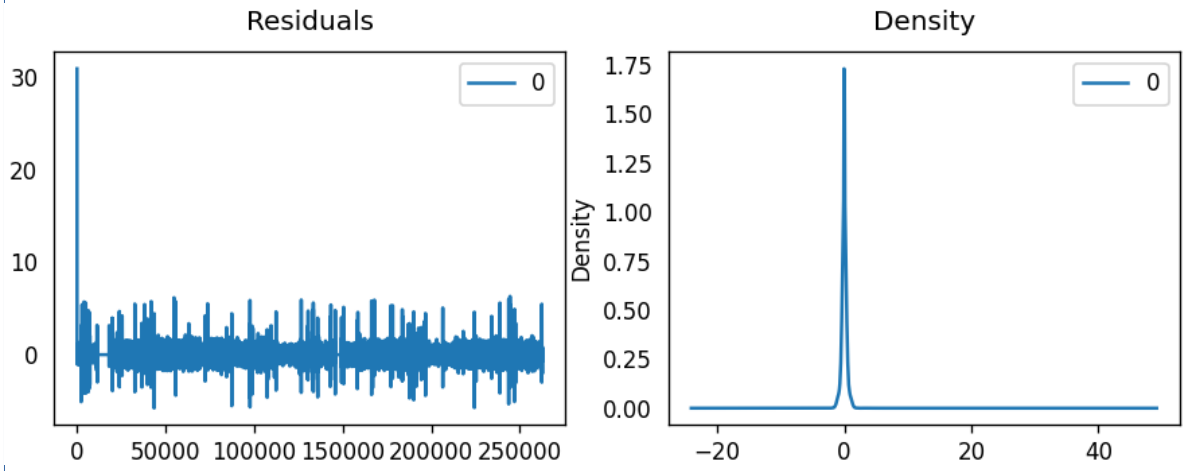




ARIMA – Ajustando o modelo

- Verificar os pesos (coef) e resíduos para avaliar modelo
- Resíduos pequenos significa que o modelo é bom

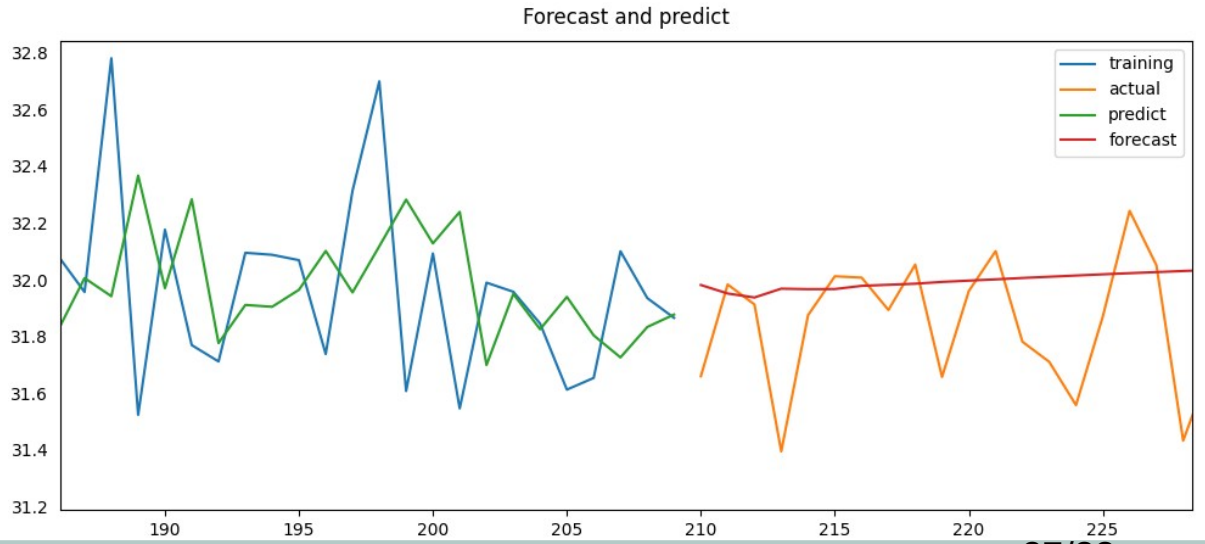
```
from statsmodels.tsa.arima.model import ARIMA

datasetsize = len(df3)
tsize = (int)(len(df3) * 0.7)
print(f'Using {tsize} of {datasetsize} samples for training')

dftrain = df3[:tsize]
dftest = df3[tsize:]

# 1,1,2 ARIMA Model
model = ARIMA(dftrain, order=(1,1,0))
model_fit = model.fit()
print(model_fit.summary())

# In[] Plot residual errors
residuals = pd.DataFrame(model_fit.resid)
fig, ax = plt.subplots(1,2)
residuals.plot(title="Residuals", ax=ax[0])
residuals.plot(kind='kde', title='Density', ax=ax[1])
plt.show()
```



ARIMA – Predição

- Usa-se método predict()
- Prever as próximas *n* amostras (n = 20 no exemplo)

```
fcp = model_fit.predict()

fc = model_fit.forecast(20, alpha=0.05)  # 95% conf

# Plot
plt.figure(figsize=(12,5), dpi=100)
plt.plot(dftrain, label='training')
plt.plot(dftest, label='actual')
plt.plot(fcp, label='predict')
plt.plot(fc, label='forecast')
plt.title('Forecast and predict')
plt.legend()
plt.show()
```



SARIMA – Incluir sasonalidade

- Em vez de inferir das amostras anteriores, infere do período anterior
- O auto_arima, da biblioteca pmdarima, ajuda a determinar os parâmetros ideais para o sarima

```
fcp = model_fit.predict()

fc = model_fit.forecast(20, alpha=0.05)  # 95% conf

# Plot
plt.figure(figsize=(12,5), dpi=100)
plt.plot(dftrain, label='training')
plt.plot(dftest, label='actual')
plt.plot(fcp, label='predict')
plt.plot(fc, label='forecast')
plt.title('Forecast and predict')
plt.legend()
plt.show()
```

Predição com LSTM

- Preparar dados: normalizar, partir dataset, aplicar janela deslizante

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature_range=(0, 1))
dataset = scaler.fit_transform(dataset)
train_size = int(len(dataset) * 0.80)
test_size = len(dataset) - train_size
train, test = dataset[0:train_size,:], dataset[train_size:len(dataset),:]

def create_dataset(dataset, look_back=1):
    X, Y = [], []
    for i in range(len(dataset)-look_back-1):
        a = dataset[i:(i+look_back), 0]
        X.append(a)
        Y.append(dataset[i + look_back, 0])
    return np.array(X), np.array(Y)

look_back = 30
X_train, Y_train = create_dataset(train, look_back)
X_test, Y_test = create_dataset(test, look_back)
# reshape input to be [samples, time steps, features]
X_train = np.reshape(X_train, (X_train.shape[0], 1, X_train.shape[1]))
X_test = np.reshape(X_test, (X_test.shape[0], 1, X_test.shape[1]))
```

Predição com LSTM

- Compilar modelo, treinar

```
from keras.models import Sequential
from keras.layers import LSTM, Dropout, Dense
from keras.callbacks import EarlyStopping

model = Sequential()
model.add(LSTM(100, input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dropout(0.2))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')

history = model.fit(X_train, Y_train, epochs=20, batch_size=70, validation_data=(X_test, Y_test),
                    callbacks=[EarlyStopping(monitor='val_loss', patience=10)], verbose=1, shuffle=False)

model.summary()
```

Predição com LSTM

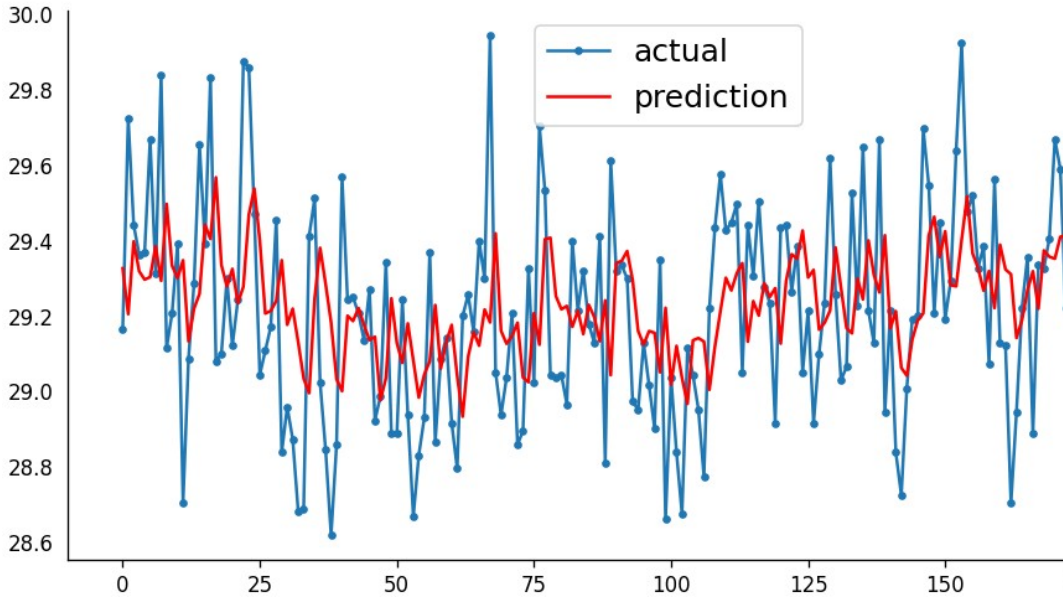
- Avaliar performance

```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error

train_predict = model.predict(X_train)
test_predict = model.predict(X_test)

# invert predictions
train_predict = scaler.inverse_transform(train_predict)
Y_train = scaler.inverse_transform([Y_train])

test_predict = scaler.inverse_transform(test_predict)
Y_test = scaler.inverse_transform([Y_test])

print('Train Mean Absolute Error:', mean_absolute_error(Y_train[0], train_predict[:,0]))
print('Train Root Mean Squared Error:',np.sqrt(mean_squared_error(Y_train[0], train_predict[:,0])))
print('Test Mean Absolute Error:', mean_absolute_error(Y_test[0], test_predict[:,0]))
print('Test Root Mean Squared Error:',np.sqrt(mean_squared_error(Y_test[0], test_predict[:,0])))
```

Predição com LSTM

- Avaliar performance

```
aa=[x for x in range(200)]
plt.figure(figsize=(8,4))
plt.plot(aa, Y_test[0][:200], marker='.', label="actual")
plt.plot(aa, test_predict[:,0][:200], 'r', label="prediction")
# plt.tick_params(left=False, labelleft=True) #remove ticks
plt.tight_layout()
sns.despine(top=True)
plt.subplots_adjust(left=0.07)
plt.ylabel('Global_active_power', size=15)
plt.xlabel('Time step', size=15)
plt.legend(fontsize=15)
plt.show();
```



Referências úteis

- https://www.statsmodels.org/stable/generated/statsmodels.tsa.arima.model.ARIMA.html
- https://www.machinelearningplus.com/time-series/time-series-analysis-python/
- https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/
- https://www.pluralsight.com/guides/advanced-time-series-modeling-(arima)-models-in-python
- https://towardsdatascience.com/time-series-analysis-visualization-forecasting-with-lstm-77a905180eba
- https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/