Métricas de avaliação na prática

Aula	16
Ready	☑
Finished	☑
Ciclos	Ciclo 02: Fundamentos

Objetivo da Aula:

□ Métricas de avaliação na prática

Conteúdo:

▼ 1. Métricas na prática com Python

Os comandos em Python para determinar as métricas de desempenho podem ser simplificadas pelo uso da biblioteca numpy.

▼ 1.1. Código em Python

```
errors = df["Predictions"] - df["Actuals"]
rmse = np.sqrt( np.mean( errors ** 2 ) )
mae = np.mean( np.abs( errors ) )
mape = np.mean( np.abs( errors / df["Actuals"] ) ) * 100

# calculo das métricas AIC e BIC
log_likelihood = model_fit.llf
num_params = model_fit.params.shape[0]
num_obs = len( train_series )
```

```
aic = -2 * log_likelihood + 2 * num_params
bic = -2 * log_likelihood + num_params * np.log( num_obs )

print( f"RMSE: {rmse}" )
print( f"MAE: {mae}" )
print( f"MAPE: {mape}" )
print( f"AIC: {aic}" )
print( f"BIC: {bic}" )
```

1.2. Código Completo até o momento

```
import numpy as np
import seaborn as sns
import pandas as pd
import warnings

from matplotlib import pyplot as plt
from statsmodels.tsa.ar_model import AutoReg as AR

# configuracao da series
np.random.seed(42)
n = 500

# criacao da series perfeita
trend = np.linspace( 0, 0, n)
noise = np.random.normal( 0, 1, n)
serie_perfeita = trend + noise

dates = pd.date_range( start='2023-01-01', periods=n, freq='D' )
serie_perfeita = pd.Series( serie_perfeita, index=dates, name='serie_perfe

# Quebra Premissa 1: Linearidade
trend_break = np.linspace( 0, 10, n )
serie_nao_linear = serie_perfeita + trend_break
```

```
# Quebra Premissa 2: Estacionariedade
seasonality = 3 * np.sin( 2 * np.pi * np.arange( n ) / 50  )
serie_nao_estacionaria = serie_perfeita + seasonality

# Quebra Premissa 3: Autocorrelacao dos residucos
autoregressive = np.zeros( n)
autoregressive[0] = noise[0]
for t in range( 1, n ):
    autoregressive[t] = 0.8 * autoregressive[t-1] + np.random.normal( 0, 0.5

serie_nao_autocorrelacao = serie_perfeita + autoregressive

# Quebra Premissa 4: Homoscedasticidade
non_normal = noise * np.linspace( 1, 3, n )
serie_nao_homoscedastica = serie_perfeita + non_normal

# Quebra Premissa 5: Não nomralidade dos residuos
non_normal_noise = np.random.exponential( scale=1, size= n )
serie_nao_normal = trend + non_normal_noise

# Combinar as series
serie_final = (
    serie_perfeita
    + trend_break
    + seasonality
    + autoregressive
    + non_normal
    + non_normal_noise
)

serie_final = pd.Series( serie_final, index=dates, name='serie_final' )
serie_final = serie_final - serie_final.min() + 1

# Visualizacao das series
plt.figure( figsize=( 16, 8 ) )
sns.lineplot( serie_final )
```

```
# Divisão da Série em Conjuntos de Treinamento, Validação e Teste
train_size = int( 0.8 * len( serie_final ) )
validation_size = int( 0.1 * len( serie_final ) )

train = serie_final[:train_size]
validation = serie_final[train_size:train_size + validation_size]
test = serie_final[train_size + validation_size:]

warnings.filterwarnings( "ignore" )

predictions = []
actuals = []
train_series = train.copy()
for t in range( len( validation ) ):
    # Training Model
    model = AR( train_series, lags=4 )
    model_fit = model.fit()

    # Prediction
    forecast = model_fit.forecast( steps=1 ).iloc[0]

    # Save Prediction
    predictions.append( forecast )
    actuals.append( validation.iloc[t] )

    # Update training dataset
    train_series = pd.concat( [train_series, pd.Series( validation.iloc[t], inde

df = pd.DataFrame( {"Predictions": predictions, "Actuals": actuals}, index=

# compute performance metrics
errors = df["Predictions"] - df["Actuals"]
rmse = np.sqrt( np.mean( errors ** 2 ) )
mae = np.mean( np.abs( errors ) )
mape = np.mean( np.abs( errors / df["Actuals"] ) ) * 100
```

```
# calculo das métricas AIC e BIC
log_likelihood = model_fit.llf
num_params = model_fit.params.shape[0]
num_obs = len( train_series )

aic = -2 * log_likelihood + 2 * num_params
bic = -2 * log_likelihood + num_params * np.log( num_obs )

print( f"RMSE: {rmse}" )
print( f"MAE: {mae}" )
print( f"MAPE: {mape}" )
print( f"AIC: {aic}" )
print( f"BIC: {bic}" )

# Plot figure
plt.figure( figsize=( 10, 5 ) )
plt.plot( train, label="Train", color="black" )
plt.plot( validation, label="Validation", color="blue" )
plt.plot( df['Predictions'], label="Previsões", color="red", linestyle="--" )
```