Belajar Peramalan Menggunakan Python - Dari Dasar Hingga Mahir

# 1. Persiapan Awal

Install terlebih dahulu beberapa library yang dibutuhkan:

pip install pandas numpy matplotlib statsmodels sklearn tensorflow

# 2. Dataset

Gunakan dataset yang berbentuk time series, misalnya data penjualan bulanan. Anda bisa menggunakan dataset dari Kaggle atau dataset bawaan library seperti pandas. Contoh data yang digunakan di sini adalah data penjualan bulanan yang tersimpan dalam file CSV sales\_data.csv.

# 3. Tahapan Peramalan

## A. Forecasting Dasar: Rata-rata

Langkah pertama dalam forecasting adalah menggunakan Average (Rata-rata) sebagai metode paling dasar untuk peramalan. Misalnya, kita ingin meramalkan penjualan bulan berikutnya menggunakan rata-rata penjualan dari bulan sebelumnya.

import pandas as pd  
data = pd.read\_csv('sales\_data.csv', parse\_dates=['Month'], index\_col='Month')  
forecast = data['Sales'].mean()  
print(f"Perkiraan penjualan bulan berikutnya: {forecast}")

## B. Forecasting Moving Average (Rata-rata Bergerak)

Moving Average adalah salah satu teknik yang lebih baik dibandingkan rata-rata sederhana, karena mempertimbangkan nilai rata-rata dalam rentang waktu tertentu.

window = 3 # Rentang waktu untuk rata-rata bergerak  
data['Moving\_Average'] = data['Sales'].rolling(window=window).mean()  
print(data.tail())

## C. Exponential Smoothing (Perataan Eksponensial)

Exponential Smoothing memberi bobot yang lebih besar pada data terbaru, sehingga cocok untuk data yang cenderung berubah.

from statsmodels.tsa.holtwinters import SimpleExpSmoothing  
model = SimpleExpSmoothing(data['Sales'])  
model\_fit = model.fit(smoothing\_level=0.2, optimized=False)  
forecast = model\_fit.forecast(steps=12)  
print(forecast)

## D. Model ARIMA (Auto-Regressive Integrated Moving Average)

ARIMA adalah salah satu model forecasting time series yang paling umum digunakan. Model ini menggabungkan komponen autoregresi, integrasi, dan moving average.

from statsmodels.tsa.arima.model import ARIMA  
model = ARIMA(data['Sales'], order=(5,1,0)) # (p,d,q)  
model\_fit = model.fit()  
forecast = model\_fit.forecast(steps=12)  
print(forecast)

## E. Forecasting dengan Model LSTM (Long Short-Term Memory)

Untuk forecasting yang lebih kompleks, model berbasis deep learning seperti LSTM sangat cocok untuk memprediksi data time series. Berikut adalah implementasi sederhana menggunakan LSTM.

### 1. Preprocessing Data

LSTM membutuhkan data yang tereskalasi (scaled) dan berbentuk sequence.

import numpy as np  
from sklearn.preprocessing import MinMaxScaler  
from tensorflow.keras.models import Sequential  
from tensorflow.keras.layers import LSTM, Dense  
  
data\_values = data['Sales'].values.reshape(-1, 1)  
scaler = MinMaxScaler(feature\_range=(0, 1))  
scaled\_data = scaler.fit\_transform(data\_values)  
train\_size = int(len(scaled\_data) \* 0.8)  
train\_data = scaled\_data[:train\_size]  
test\_data = scaled\_data[train\_size:]  
  
def create\_dataset(data, time\_step=1):  
 X, y = [], []  
 for i in range(len(data) - time\_step - 1):  
 X.append(data[i:(i + time\_step), 0])  
 y.append(data[i + time\_step, 0])  
 return np.array(X), np.array(y)  
  
time\_step = 3  
X\_train, y\_train = create\_dataset(train\_data, time\_step)  
X\_test, y\_test = create\_dataset(test\_data, time\_step)  
  
X\_train = X\_train.reshape(X\_train.shape[0], time\_step, 1)  
X\_test = X\_test.reshape(X\_test.shape[0], time\_step, 1)

### 2. Membangun dan Melatih Model LSTM

Membangun model LSTM sederhana menggunakan keras dan tensorflow.

model = Sequential()  
model.add(LSTM(50, return\_sequences=True, input\_shape=(time\_step, 1)))  
model.add(LSTM(50, return\_sequences=False))  
model.add(Dense(1))  
  
model.compile(optimizer='adam', loss='mean\_squared\_error')  
model.fit(X\_train, y\_train, epochs=100, batch\_size=32, verbose=1)  
  
train\_predict = model.predict(X\_train)  
test\_predict = model.predict(X\_test)  
  
train\_predict = scaler.inverse\_transform(train\_predict)  
test\_predict = scaler.inverse\_transform(test\_predict)  
  
import matplotlib.pyplot as plt  
  
plt.plot(scaler.inverse\_transform(scaled\_data), label='Data Asli')  
plt.plot(np.arange(time\_step, len(train\_predict) + time\_step), train\_predict, label='Prediksi Training')  
plt.plot(np.arange(len(train\_predict) + 2\*time\_step, len(train\_predict) + 2\*time\_step + len(test\_predict)), test\_predict, label='Prediksi Testing')  
plt.legend()  
plt.show()

# 4. Kesimpulan

Anda dapat mulai dari teknik peramalan yang paling sederhana seperti rata-rata, moving average, hingga metode yang lebih kompleks seperti ARIMA dan LSTM. Setelah terbiasa dengan konsep-konsep dasar, Anda dapat mengeksplorasi model lebih lanjut dan mengoptimalkan parameter yang lebih kompleks untuk meningkatkan akurasi prediksi.