Python Time Series Analysis, comprehensive Guide for Data Scientists

Created by Daniel Zaldaña https://x.com/ZaldanaDaniel

December 4, 2024

Contents

1 Framework Overview

Time Series Libraries

- Pandas: https://pandas.pydata.org/ - Data manipulation and time series tools
- StatsModels: https://www.statsmodels.org/ - Statistical models and tests
- Prophet: https://facebook.github.io/prophet/ - Forecasting at scale by Meta
- Darts: https://unit8co.github.io/darts/ - Time series made easy
- Sktime: https://www.sktime.org/ - Unified interface for ML with time series
- TensorFlow Time: https://www.tensorflow.org/ - Deep learning for time series
- pmdarima: https://alkaline-ml.com/pmdarima/ - ARIMA models made easy
- KATS: https://github.com/facebookresearch/kats - Time series analysis by Meta

2 Data Manipulation with Pandas

2.1 Time Series Objects

Creating Time Series

```python
# Create time series from dates
dates = pd.date_range('2023-01-01',
                      periods=10,
                      freq='D')
ts = pd.Series(data, index=dates)

# Convert column to datetime
df['date'] = pd.to_datetime(df['date'])
```

2.2 Resampling and Rolling Windows

```python
# Resample to monthly frequency
monthly = ts.resample('M').mean()
```

```
# Calculate rolling mean
rolling_mean = ts.rolling(window=7).mean()
```

3 Framework Examples

3.1 Pandas Time Series

Advanced Pandas Operations

```
import pandas as pd

# Create datetime index
dates = pd.date_range(start='2023-01-01',
                      end='2023-12-31',
                      freq='D')

# Time series operations
ts = pd.Series(np.random.randn(len(dates)),
               index=dates)

# Datetime accessors
ts.dt.year
ts.dt.month
ts.dt.day_name()

# Shifting and lagging
ts.shift(periods=1)  # Lag
ts.shift(periods=-1)  # Lead

# Rolling statistics
ts.rolling(window=7).mean()
ts.rolling(window=7).std()
ts.rolling(window=7).quantile(0.95)

# Resampling
ts.resample('M').mean()  # Monthly
ts.resample('Q').sum()   # Quarterly
ts.resample('Y').last()  # Yearly
```

3.2 StatsModels Examples

Statistical Time Series Analysis

```
import statsmodels.api as sm
from statsmodels.tsa.seasonal import STL

# Decomposition using STL
stl = STL(ts, period=12)
result = stl.fit()
seasonal = result.seasonal
trend = result.trend
resid = result.resid

# SARIMAX Model
model = sm.tsa.statespace.SARIMAX(
    ts,
    order=(1, 1, 1),
    seasonal_order=(1, 1, 1, 12),
    enforce_stationarity=False
)
results = model.fit()

# Diagnostics
results.plot_diagnostics()
print(results.summary())

# Forecast
forecast = results.get_forecast(steps=30)
conf_int = forecast.conf_int()
```

3.3 Prophet Advanced Usage

Meta Prophet Examples

```python
from prophet import Prophet

# Advanced Prophet model
model = Prophet(
    changepoint_prior_scale=0.05,
    seasonality_prior_scale=10,
    holidays_prior_scale=10,
    seasonality_mode='multiplicative',
    changepoint_range=0.9,
    yearly_seasonality=True,
    weekly_seasonality=True,
    daily_seasonality=False
)

# Add custom seasonality
model.add_seasonality(
    name='monthly',
    period=30.5,
    fourier_order=5
)

# Add country holidays
model.add_country_holidays(country_name='
    US')

# Fit and predict
model.fit(df)
future = model.make_future_dataframe(
    periods=365,
    freq='D',
    include_history=True
)
forecast = model.predict(future)

# Components plot
model.plot_components(forecast)
```

3.4 Darts Framework

Darts Time Series Tools

```python
from darts import TimeSeries
from darts.models import (
    Prophet,
    ARIMA,
    ExponentialSmoothing,
    TCNModel,
    TransformerModel
)

# Create Darts TimeSeries
series = TimeSeries.from_dataframe(
    df,
    'date',
    'value',
    freq='D'
)

# Split data
train, val = series.split_before(0.8)

# Multiple models comparison
models = [
    Prophet(),
    ARIMA(p=2, d=1, q=1),
    ExponentialSmoothing(),
    TCNModel(
        input_chunk_length=24,
        output_chunk_length=12
    ),
    TransformerModel(
        input_chunk_length=24,
        output_chunk_length=12
    )
]

# Fit and evaluate
for model in models:
    model.fit(train)
    forecast = model.predict(len(val))
    print(f"MAPE: {mape(val, forecast)}")
```

3.5 Sktime Examples

Sktime Unified Interface

```python
from sktime.forecasting.base import
    ForecastingHorizon
from sktime.forecasting.compose import
    TransformedTargetForecaster
from sktime.forecasting.theta import
    ThetaForecaster
from sktime.transformations.series.
    detrend import Deseasonalizer

# Create pipeline
forecaster = TransformedTargetForecaster
    ([
     ('deseasonalize', Deseasonalizer()),
     ('forecast', ThetaForecaster())
])

# Fit and predict
forecaster.fit(y_train)
fh = ForecastingHorizon(y_test.index,
    is_relative=False)
y_pred = forecaster.predict(fh)

# Cross-validation
from sktime.forecasting.model_selection
    import (
     temporal_train_test_split,
     SingleWindowSplitter
)

cv = SingleWindowSplitter(
     train_length=100,
     test_length=24,
     step_length=12
)
```

3.6 pmdarima (Auto ARIMA)

Automated ARIMA Modeling

```python
import pmdarima as pm

# Auto ARIMA
model = pm.auto_arima(
     y,
     start_p=1,
     start_q=1,
     test='adf',
     max_p=3,
     max_q=3,
     m=12,
     start_P=0,
     seasonal=True,
     d=1,
     D=1,
     trace=True,
     error_action='ignore',
     suppress_warnings=True,
     stepwise=True
)

# Get model summary
print(model.summary())

# Make predictions
n_periods = 24
forecast, conf_int = model.predict(
     n_periods=n_periods,
     return_conf_int=True
)

# Update model with new data
model.update(new_data)
```

3.7 KATS (by Meta)

Meta KATS Framework

```
from kats.consts import TimeSeriesData
from kats.models.prophet import ProphetModel
from kats.models.sarima import SARIMAModel
from kats.detectors.outlier import OutlierDetector
from kats.tsfeatures.tsfeatures import TsFeatures

# Create TimeSeriesData object
ts = TimeSeriesData(
    df[['time', 'value']]
)

# Detect outliers
outlier_detector = OutlierDetector(ts)
outliers = outlier_detector.detector()

# Calculate time series features
features = TsFeatures().transform(ts)

# Prophet model with KATS
params = ProphetModel.get_parameter_search_space()
model = ProphetModel(ts, params)
model.fit()
forecast = model.predict(steps=30)

# SARIMA model with KATS
sarima = SARIMAModel(
    ts,
    p=2,
    d=1,
    q=1,
    seasonal_p=1,
    seasonal_d=1,
    seasonal_q=1,
    seasonal_period=12
)
sarima.fit()
sarima_forecast = sarima.predict(steps=30)
```

4 Statistical Analysis

4.1 Decomposition

Time Series Components

```
from statsmodels.tsa.seasonal import seasonal_decompose

result = seasonal_decompose(ts,
    model='multiplicative',
    period=12)
trend = result.trend
seasonal = result.seasonal
residual = result.resid
```

4.2 Stationarity Tests

```
from statsmodels.tsa.stattools import adfuller

def check_stationarity(ts):
    result = adfuller(ts)
    print(f'ADF Statistic: {result[0]}')
    print(f'p-value: {result[1]}')
```

5 Forecasting Models

5.1 Prophet

Facebook Prophet

```
from prophet import Prophet

# Initialize and fit
model = Prophet(
    changepoint_prior_scale=0.05,
    seasonality_prior_scale=10,
    seasonality_mode='multiplicative'
)
model.fit(df)

# Make predictions
future = model.make_future_dataframe(
    periods=30
)
forecast = model.predict(future)
```

5.2 SARIMA Models

```
from statsmodels.tsa.statespace.sarimax import SARIMAX

model = SARIMAX(ts,
    order=(1, 1, 1),
    seasonal_order=(1, 1, 1, 12)
)
results = model.fit()
```

6 Deep Learning Approaches

6.1 TensorFlow Time Series

LSTM for Time Series

```
from tensorflow.keras.models import
    Sequential
from tensorflow.keras.layers import LSTM

model = Sequential([
    LSTM(50, activation='relu',
         input_shape=(n_steps, n_features
         )),
    Dense(1)
])
model.compile(optimizer='adam',
              loss='mse')
```

6.2 PyTorch Forecasting

```
from pytorch_forecasting import
    TimeSeriesDataSet
from pytorch_forecasting import
    TemporalFusionTransformer

# Create dataset
training = TimeSeriesDataSet(
    data=training_data,
    time_idx="time_idx",
    target="target",
    group_ids=["group"],
    max_encoder_length=24,
    max_prediction_length=12
)
```

7 Visualization Techniques

7.1 Matplotlib/Seaborn

Time Series Plots

```
import seaborn as sns
import matplotlib.pyplot as plt

# Time series line plot
plt.figure(figsize=(12, 6))
sns.lineplot(data=df, x='date', y='value'
    )

# Multiple time series
sns.lineplot(data=df, x='date', y='value'
    ,
            hue='category')
```

8 Best Practices

Common Pitfalls to Avoid:

- Not checking for stationarity
- Ignoring seasonality
- Data leakage in train/test split
- Not handling missing values properly

8.1 Performance Metrics

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

mae = mean_absolute_error(y_true, y_pred)
rmse = np.sqrt(mean_squared_error(
    y_true, y_pred))
mape = np.mean(np.abs((y_true - y_pred)
    / y_true)) * 100
```

9 Advanced Topics

9.1 Feature Engineering

Time-Based Features

```
# Extract datetime components
df['year'] = df['date'].dt.year
df['month'] = df['date'].dt.month
df['day'] = df['date'].dt.day
df['dayofweek'] = df['date'].dt.dayofweek

# Lag features
df['lag_1'] = df['value'].shift(1)
df['lag_7'] = df['value'].shift(7)

# Rolling features
df['rolling_mean'] = df['value'].rolling(
    window=7).mean()
df['rolling_std'] = df['value'].rolling(
    window=7).std()
```