statsmodel

Time Series Analysis

Package Overview

- A Python package oriented towards data analysis, data science, and statistics
- Allows users to explore data, estimate statistical models, and perform statistical tests
- Integrates with Pandas for handling data
- Recommended to install using conda
- Contributions are welcomed
- Bug reports are handled through an issue tracker on GitHub

Package History

- Originally *statsmodels* was just *models* and was a module in scipy.stats
- *models* was written by Jonathan Taylor (https://statweb.stanford.edu/~jtaylo/index.html)
- After *statsmodels* was corrected and tested at the 2009 Google Summer of Code, *statsmodels* was released as its own package
- Continued development since then and regularly adds new models, plotting tools, and statistical methods

statsmodels.tsa

- Classes and functions assist users with:
  - Descriptive Statistics & Tests
  - Estimation
  - Autoregressive-moving-average (ARMA) Process
  - Autoregressive Distributed Lag (ARDL) Models
  - Error Correction Models (ECM)
  - Regime Switching Models
  - Time Series Filters
  - TSA Tools
  - VARMA Process
  - Interpolation
  - Deterministic Processes
  - Forecasting Models
  - Prediction Results

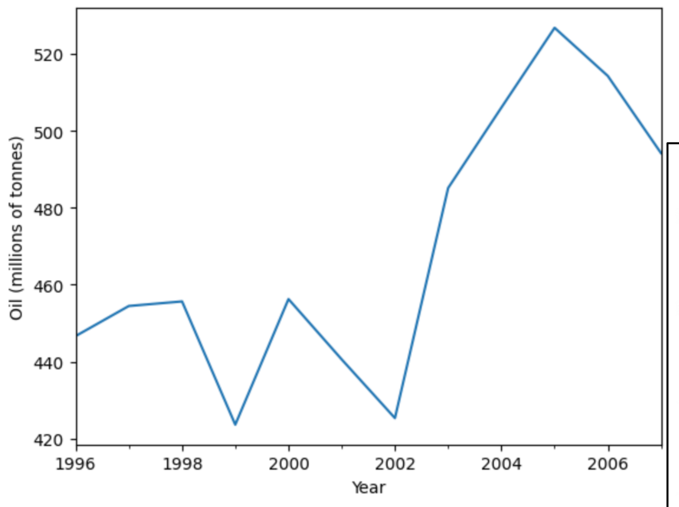
An example provided on Statsmodels website shows how a user can take data, in this case, oil production in Saudi Arabia, use statsmodels.tsa methods to smooth the data and then forecast additional data.

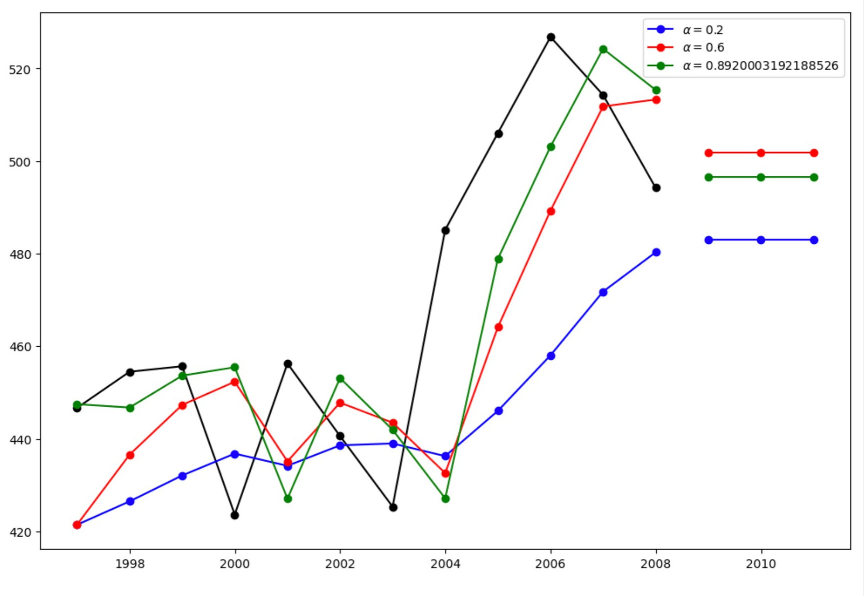
Example

Simple Exponential Smoothing

```
ax = oildata.plot()
ax.set_xlabel("Year")
ax.set_ylabel("Oil (millions of tonnes)")
print("Figure 7.1: Oil production in Saudi Arabia from 1996 to 2007.")
```

Figure 7.1: Oil production in Saudi Arabia from 1996 to 2007.






```
fit1 = SimpleExpSmoothing(oildata, initialization_method="heuristic").fit(
    smoothing_level=0.2, optimized=False
)
fcast1 = fit1.forecast(3).rename(r"$\alpha=0.2$")
fit2 = SimpleExpSmoothing(oildata, initialization_method="heuristic").fit(
    smoothing_level=0.6, optimized=False
)
fcast2 = fit2.forecast(3).rename(r"$\alpha=0.6$")
fit3 = SimpleExpSmoothing(oildata, initialization_method="estimated").fit()
fcast3 = fit3.forecast(3).rename(r"$\alpha=%s$" % fit3.model.params["smoothing_level"])

plt.figure(figsize=(12, 8))
plt.plot(oildata, marker="o", color="black")
plt.plot(fit1.fittedvalues, marker="o", color="blue")
(line1,) = plt.plot(fcast1, marker="o", color="blue")
plt.plot(fit2.fittedvalues, marker="o", color="red")
(line2,) = plt.plot(fcast2, marker="o", color="red")
plt.plot(fit3.fittedvalues, marker="o", color="green")
(line3,) = plt.plot(fcast3, marker="o", color="green")
plt.legend([line1, line2, line3], [fcast1.name, fcast2.name, fcast3.name])
```

In this example, the optimization offered through statsmodels was not utilized in the first two models and the smoothing level was experimented with to compare results in the graph above-right.

Exponential Smoothing

- Exponential smoothing is a method of forecasting that weights more recent data heavier than older data. The data’s weight has an exponential decay to its weight as it ages.
- Exponential smoothing does lag a period because it uses the data from the last period. It also doesn’t handle trends well and is best used for short-term forecasts that don’t reflect seasonality or cycles

Question?

How efficient are the models in this package and are there more efficient alternatives?

Experiment Brainstorming

Idea 1

Look at global emissions for certain time period before and after the pandemic lockdowns

- Plot the data time x emissions
- Determine if linear or non-linear
- Look for trends in the data
- Use weighted forecast models to see how global emissions will continue in the post-pandemic world

Experiment Brainstorming

Idea 2

Look at rainfall over certain time and overlay it with severity of forest fires. Look for forest fire severities based on rainfall forecasting.

- My thinking with this would be to analyze the data for trends or cycles between the two data sets and try to find correlations between them.

Experiment Brainstorming

Idea 3

Stock price forecasting for certain companies.

- Choose a company and get data per second, minute, hour, and day
- Analyze the data and run forecasting models for each time frame
- Compare how different models perform for the different data sets