Boussiala Mohamed Nachid

boussiala.nachid@univ-alger3.dz

Time Series Forecasting using PyCaret

PyCaret

`PyCaret` is a low-code Python library for `Machine Learning` that automates workflows and offers efficient end-to-end experiments. Automating major steps in evaluation and comparing models makes it easier for us to examine the performance of different models and select the best one. On this page, let's see how to approach a time series forecasting problem of familiar airline data with PyCaret.

Install and import PyCaret

```
pip install pycaret
import pycaret
```

Prepare data

Data available within the PyCaret library can be imported with `pycaret.datasets.get_data()`. Otherwise, import using pandas from the device as usual: `pd.read_xlsx('path')`

load Airline data

```
# Loading Sample Dataset
from pycaret.datasets import get_data
airline= get_data('airline')  # to load from pycaret library

Period
1949-01    112.0
1949-02    118.0
1949-03    132.0
1949-04    129.0
1949-05    121.0
Freq: M, Name: Number of airline passengers, dtype: float64
```

Setup Environment

The `setup()` function initializes the environment for training and creates the transformation pipeline. It is important to call `setup()` before executing any other function. For the `Time Series` experiment, the following parameters should be passed to the setup function.

1. `fh`: forecasting horizon
2. `fold`: number of folds for cross validation
3. `session_id`: random seed for experiment

Import Time Series libraries

```
#init setup
from pycaret.time_series import *
s_airline= setup(airline, fh=3, session_id= 123)
```

```
<pandas.io.formats.style.Styler at 0x27fc645f950>
```

As you can see in the above table, AutoML analyzes the time series data and provides a detailed summary, especially about **seasonality**. Have a look into entries Seasonality Present and Significant Seasonal Period(s) in the above table generated by AutoML.

Choose API and Compare Models

`PyCaret` has two API's. `Functional` and `OOP`.

```
1.Functional API follows a procedural approach and offers a
straightforward interface. Since the settings are applied globally
across functions, this API is convenient for quick experiments.
2.The OPP API follows an object-oriented approach, providing a
customizable and modular workflow and, hence, more suitable for
running multiple experiments independently.
```

In this experiment, we use functional API (Codes for OOP API are commented).

The `compare_model()` function returns a table containing different models and performance metrics. It can be observed from the table that Exponential smoothing outperforms all other models.

```
#model training and selection
#Compare models using Functional API
best_airline= compare_models()
```

```
<IPython.core.display.HTML object>
```

```
<pandas.io.formats.style.Styler at 0x27fc7a72650>
```

```
<IPython.core.display.HTML object>
```

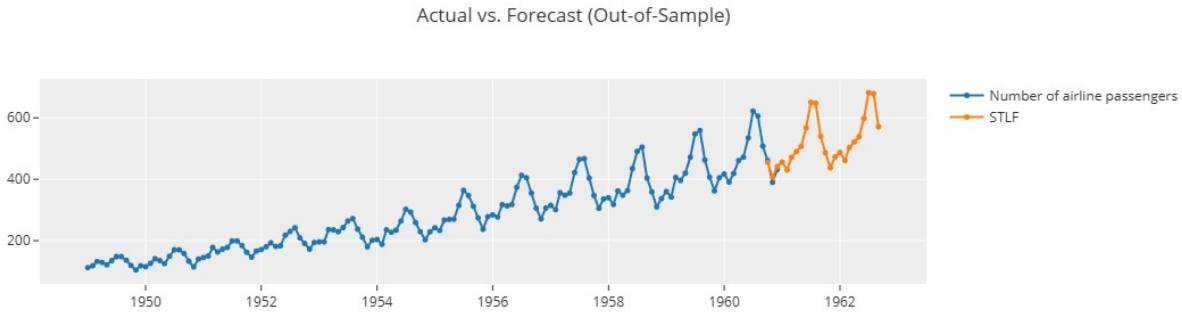
```
# '''OOP API'''
s= TSForecastingExperiment()
s.setup(airline, fh=3, session_id= 123)
best_airline_OOP = s.compare_models()
```

```
<pandas.io.formats.style.Styler at 0x27fc79cb5d0>

<IPython.core.display.HTML object>

<pandas.io.formats.style.Styler at 0x27fc74f9410>

<IPython.core.display.HTML object>
```

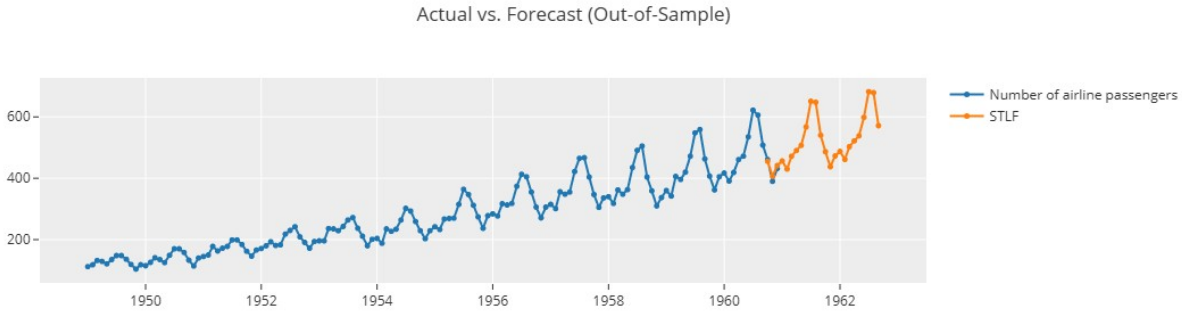
Plot the Forecast and Diagnose Residuals.

Now let's plot the forecast using `plot_model()` function. In data_kwargs the dictionary, we can also give keywords for the plot, such as title, legend, opacity, height, width etc. as per requirement.

```
'''Plot forecasting using functional API'''
plot_model(best_airline, plot = 'forecast', data_kwargs = {'fh' : 24})
```

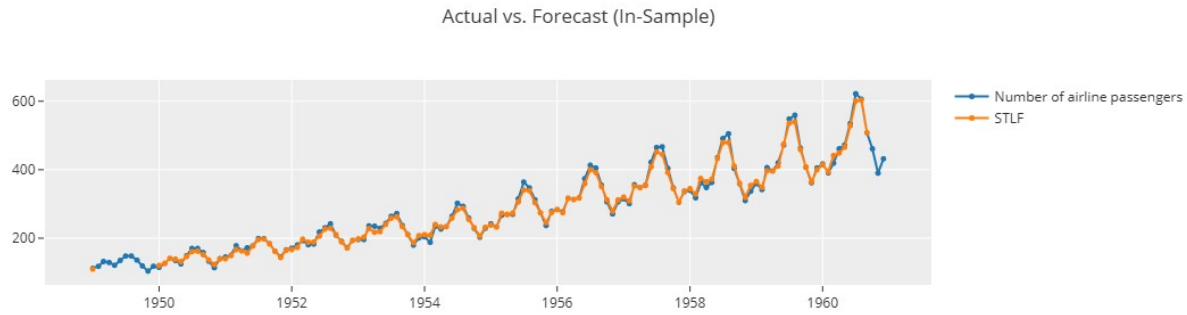



```
''' OOP API '''
s.plot_model(best_airline_OOP, plot = 'forecast', data_kwargs =
{'fh' : 24})
```

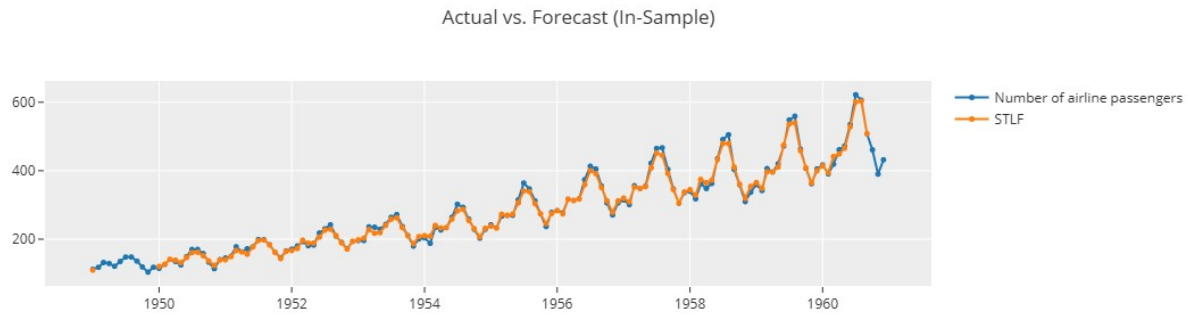


The model's prediction on training data (in sample) can be plotted as below.

```
'''Insample plot using functional API'''
plot_model(best_airline, plot = 'insample')
```




```
''' OOP API'''
s.plot_model(best_airline_OOP, plot = 'insample')
```



make a prediction on hold-out/test set

```
#predict on hold-out/test set using functional API
pred_holdout_airline= predict_model(best_airline)
```

```
<pandas.io.formats.style.Styler at 0x27fc8eeced0>
```

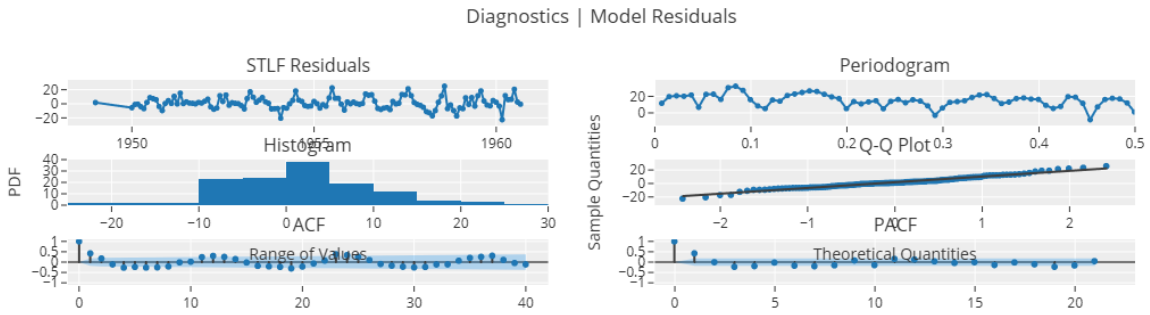
```
#predict on hold-out/test set using OOP
pred_holdout_airline= s.predict_model(best_airline_OOP)
```

```
<pandas.io.formats.style.Styler at 0x27fc7428850>
```

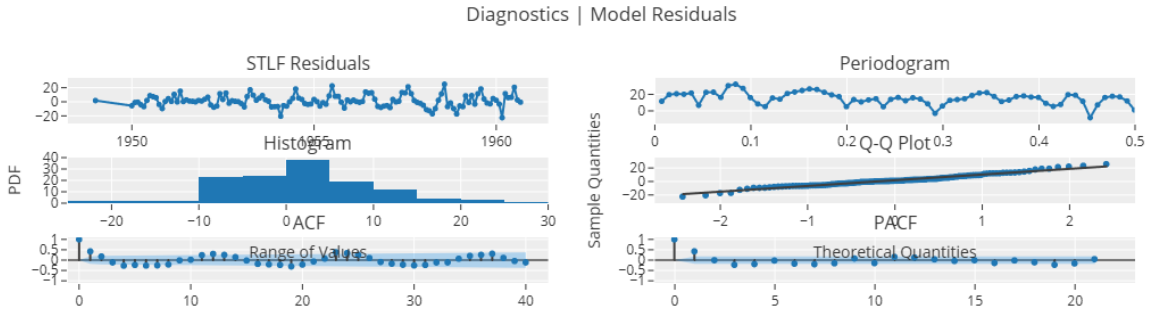
Make a Diagnostic

To diagnose the `residuals`, we can use the same `plot_model()` function with key word `plot = 'diagnostics'` as below.

```
'''Diagnose best model using functional API'''
plot_model(best_airline, plot = 'diagnostics')
```




```
'''Diagnose best model using OOP API'''
s.plot_model(best_airline_OOP, plot = 'diagnostics')
```



Or make a prediction in unseen future

```
'''predict in unseen future using functional API'''
predictions_airline= predict_model(best_airline, fh=36)

'''predict in unseen future using OOP '''
predictions_airline_OOP= predict_model(best_airline_OOP, fh=36)
```