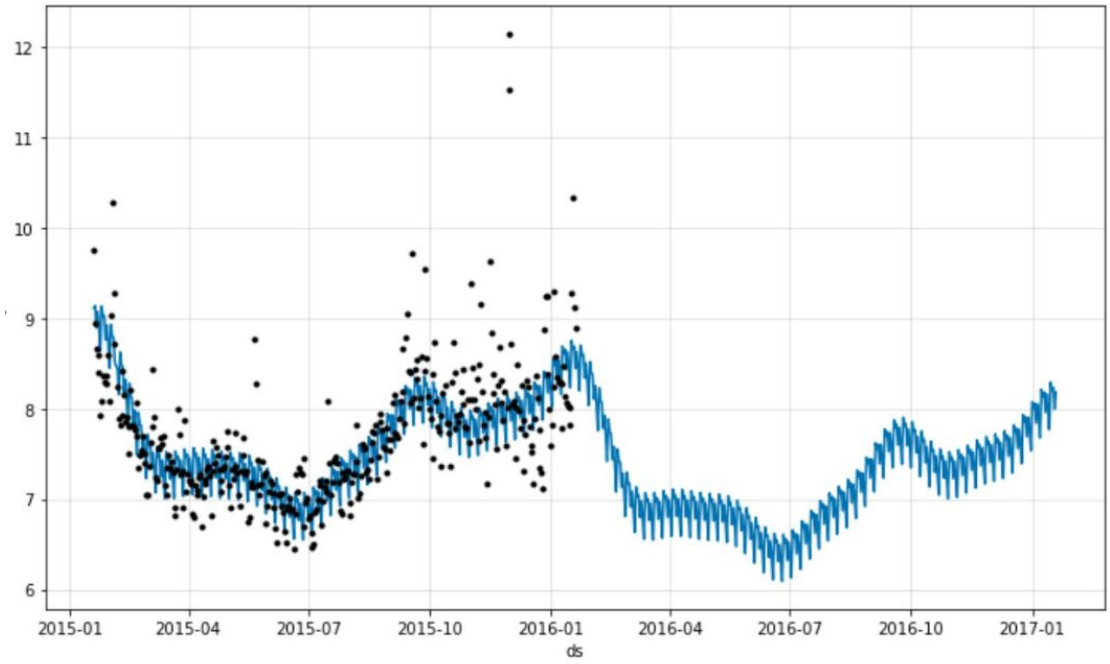
NeuralProphet A simple forecasting model based on Neural Networks in PyTorch:



NeuralProphet is a python library used for modeling time-series data based on neural networks. It helps in building forecasting models for scenarios where external factors are taken into consideration, which can drive the behaviors of the target series over time. This is heavily inspired by Prophet, which is a popular forecasting tool developed by Facebook.

NeuralProphet has a number of added features compared to the original Prophet. They are as follows:

- Gradient Descent for optimization using PyTorch as the backend makes the modelling process much faster.
- Using AR-Net for modelling time-series autocorrelation.
- Custom losses and metrics

Model Components of NeuralProphet:

- Trend
- Seasonality
- Auto-Regression
- Lagged Regressors
- Events
- Future Regressors

Installation:

Cloning github repository:

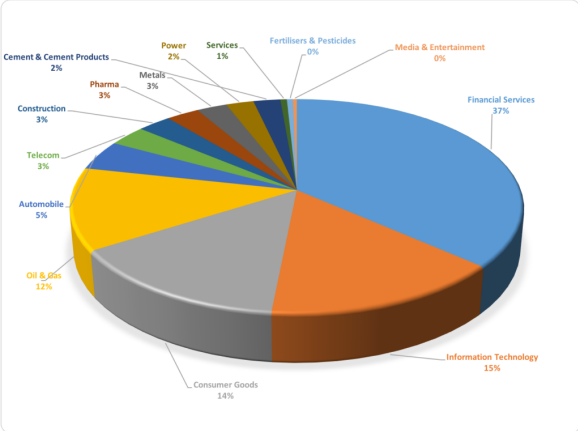
1. git clone https://github.com/ourownstory/neural_prophet
2. After downloading change to repository directory using cd neural_prophet
3. Pip install .[live]

Note: If you plan to use the package in a Jupyter notebook, it is recommended to install the 'live' package version with pip install .[live]. This will allow you to enable plot_live_loss in the train function to get a live plot of train (and validation) loss.

Implementation with Case study:

I would recommend creating a fresh working environment (conda or venv) and install NeuralProphet packages from the new environment letting the installer take care of all dependencies (Pandas, Jupyter Lab/Notebook, PyTorch).

Dataset:
For our Case study we will use the Nifty Indices Dataset.



The dataset can be downloaded here.

Context:

The National Stock Exchange of India Limited (NSE) is the leading stock exchange of India, located in Mumbai. The NIFTY 50 index is the National Stock Exchange of India's benchmark broad based stock market index for the Indian equity market.

Apart from the NIFTY 50 index, there are also other indices like NIFTY Next 50, Nifty Midcap 150 etc. Exploring these indices may help in taking investment decisions.

Content:

This dataset has day level information on major NIFTY indices starting from 01 January 2000.

Each file represents an index and has the following columns:

- Date - date of observation
- Open - open value of the index on that day
- High - highest value of the index on that day
- Low - lowest value of the index on that day
- Close - closing value of the index on that day
- Volume - volume of transaction
- Turnover - turn over
- P/E - price to earnings ratio
- P/B - price to book value
- Div Yield - dividend yield

Stock price forecasting using NeuralProphet:

```
# First import main packages

import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from neuralprophet import NeuralProphet

# Then, we can read the data into a Panda DataFrame.

data=pd.read_csv('/kaggle/input/nifty-indices-dataset/NIFTYIT.csv',parse_dates=['Date'])
data.head()
```

	Date	Open	High	Low	Close	Volume	Turnover	P/E	P/B	Div Yield
0	2000-01-03	0.0	0.0	0.0	45015.92	3777566.0	5.584100e+09	196.98	52.27	0.06
1	2000-01-04	0.0	0.0	0.0	48221.50	12128996.0	1.818670e+10	211.00	55.99	0.06
2	2000-01-05	0.0	0.0	0.0	46430.77	23400610.0	3.136010e+10	203.17	53.91	0.06
3	2000-01-06	0.0	0.0	0.0	44433.55	18408571.0	2.370960e+10	194.43	51.59	0.06
4	2000-01-07	0.0	0.0	0.0	41106.21	2762086.0	2.938500e+09	179.87	47.73	0.07

```
# Pattern for neuralprophet
```

A NeuralProphet object expects the time-series data to have a date column named **ds** and the time-series column value we want to predict as **y**.

For stocks we will use **ds** as data and **y** as ‘P/E’

```
data = data[["Date", "P/E"]]
data.rename(columns={'Date':'ds','P/E':'y'},inplace=True)

data.head()
```

	ds	y
0	2000-01-03	196.98
1	2000-01-04	211.00
2	2000-01-05	203.17
3	2000-01-06	194.43
4	2000-01-07	179.87

Now let's initialize the model

Below, I've brought all default arguments defined for the NeuralProphet object, including additional information about some. These are the hyperparameters you can configure in the model. If you are planning to use the default variables, you can just do `model = NeuralProphet()`.

For a complete Hyperparametera tuning view here.

Defining the model and fitting it

```
model = NeuralProphet()
metrics = model.fit(data, validate_each_epoch=True, freq="D")
```

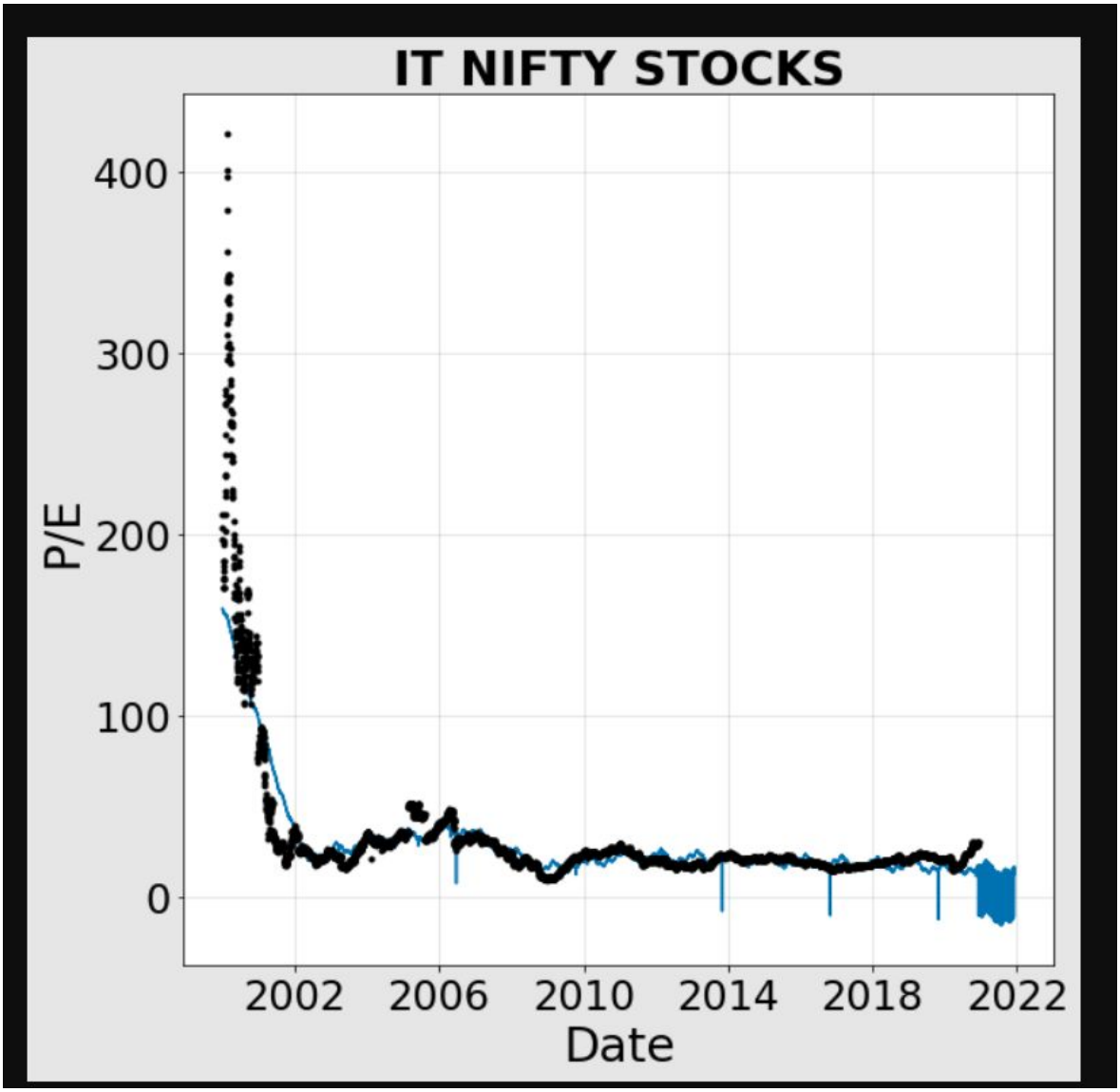
Here freq="D" is calendar day frequency

Making future prediction from the model

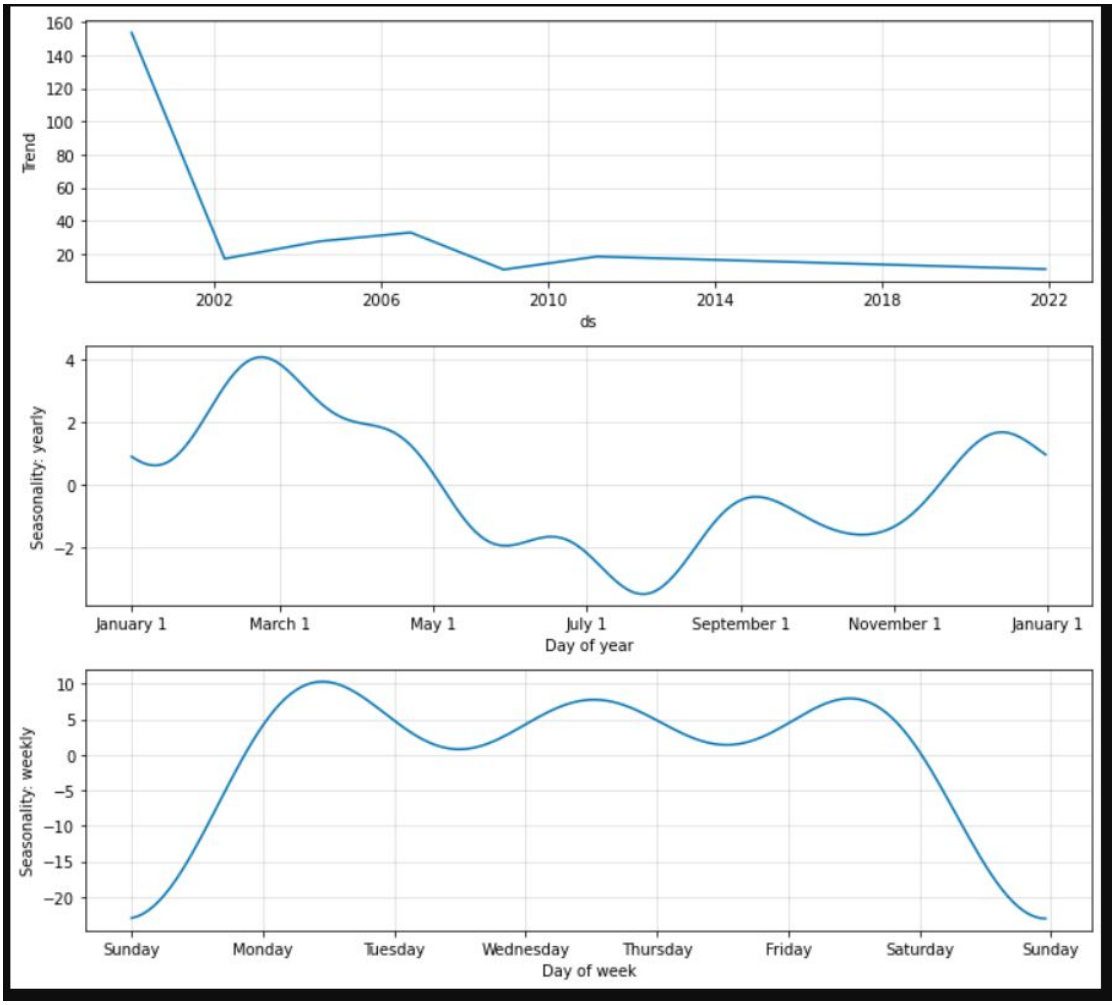
```
f= model.make_future_dataframe(data, periods=365, n_historic_predictions=len(data))
forecast = model.predict(f)
```

Plotting the prediction made by model

```
fig, ax = plt.subplots(figsize=(8,8))
model.plot(forecast, xlabel="Date", ylabel="P/E", ax=ax)
ax.xaxis.label.set_size(28)
ax.yaxis.label.set_size(28)
ax.tick_params(axis='both', which='major', labelsize=24)
ax.set_title("IT NIFTY STOCKS", fontsize=28, fontweight="bold")
```

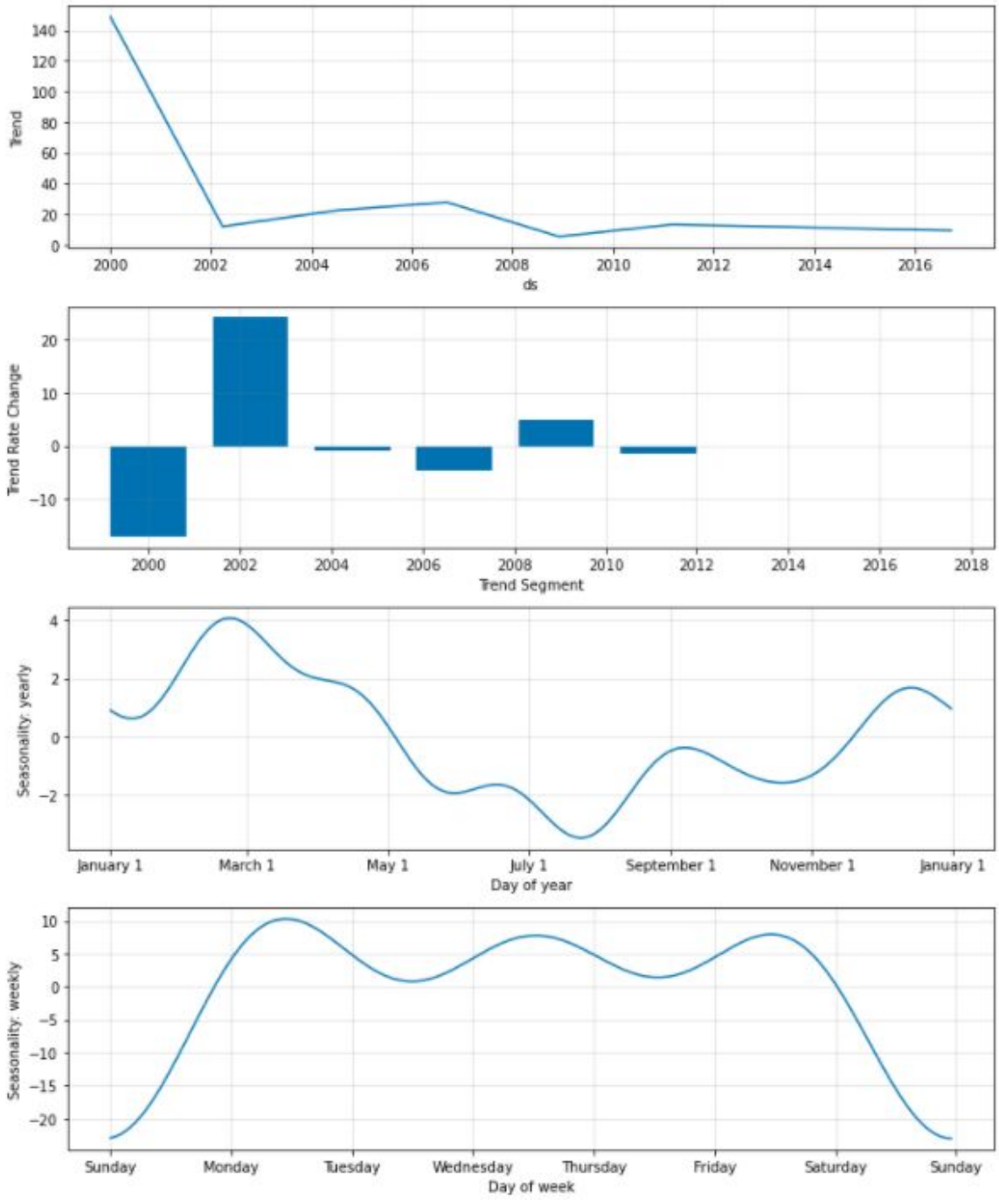


model_components = model.plot_components(forecast)



Plotting the parameters of the model

This shows Trend,Trend Rate change, weekly seasonality, yearly seasonality.

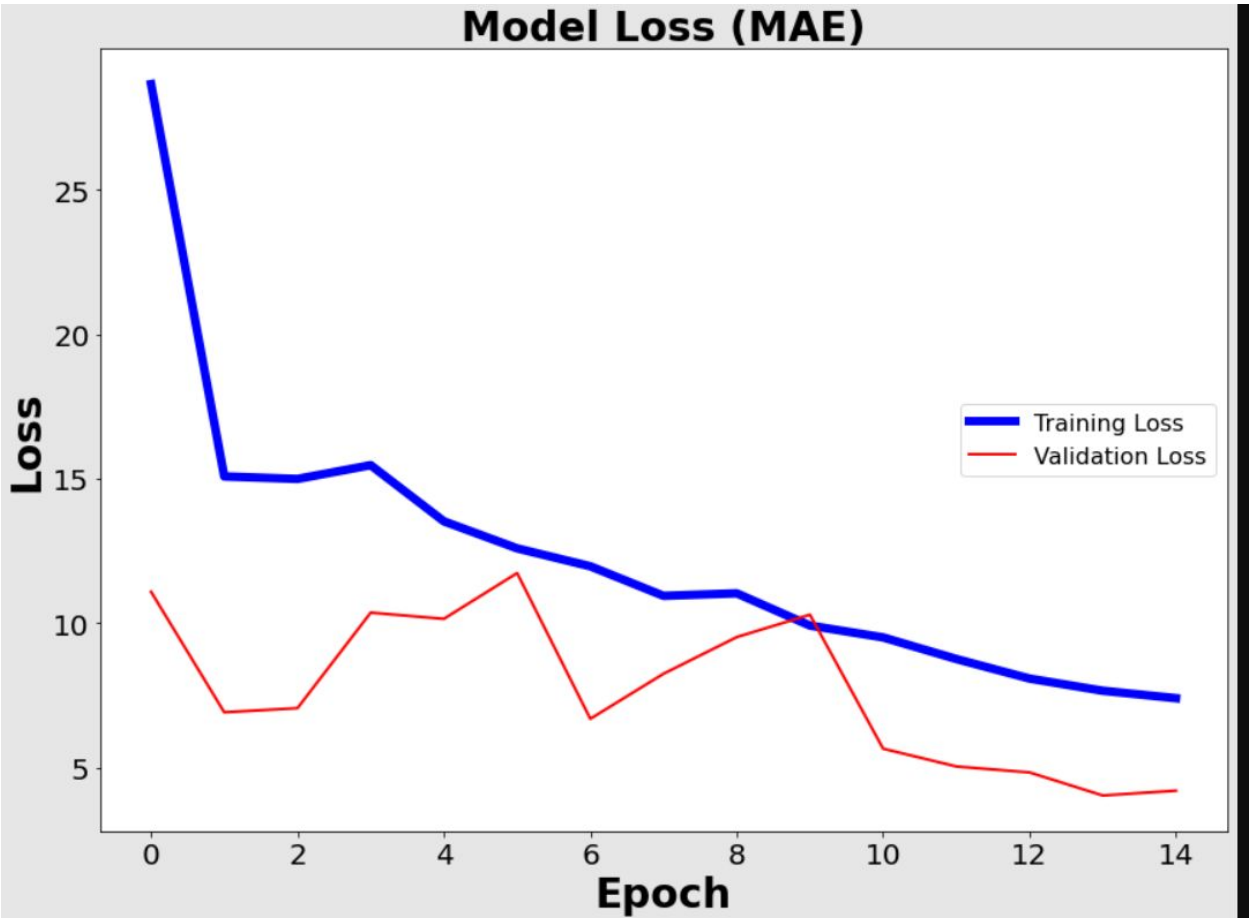


```
# Plotting Model Loss

# The model loss using Mean Absolute Error(MAE) is plotted below.

fig, ax = plt.subplots(figsize=(14, 10))
ax.plot(metrics["MAE"], '-b', linewidth=6, label="Training Loss")
ax.plot(metrics["MAE_val"], '-r', linewidth=2, label="Validation Loss")

ax.legend(loc='center right', fontsize=16)
ax.tick_params(axis='both', which='major', labelsize=20)
ax.set_xlabel("Epoch", fontsize=28, fontweight="bold")
ax.set_ylabel("Loss", fontsize=28, fontweight="bold")
ax.set_title("Model Loss (MAE)", fontsize=28, fontweight="bold")
```



The complete kaggle notebook can be accessed here.

Conclusion:

In this post, we talked about NeuralProphet, a python library that models time-series based on Neural Networks. The library uses PyTorch as a backend. As a case study, we created a prediction model NIFTY IT Stocks time-series data and made a prediction. An advantage of using this library is its similar syntax to Facebook's Prophet library.

References:

NeuralProphet,Documentation