Introduction

Nixtla's TimeGPT is a generative pre-trained forecasting model for time series data. TimeGPT can produce accurate forecasts for new time series without training, using only historical values as inputs. TimeGPT can be used across a plethora of tasks including demand forecasting, anomaly detection, financial forecasting, and more.

The TimeGPT model "reads" time series data much like the way humans read a sentence – from left to right. It looks at windows of past data, which we can think of as "tokens", and predicts what comes next. This prediction is based on patterns the model identifies in past data and extrapolates into the future.

The API provides an interface to TimeGPT, allowing users to leverage its forecasting capabilities to predict future events. TimeGPT can also be used for other time series-related tasks, such as what-if scenarios, anomaly detection, and more.

Usage

In [ ]:
```
# | hide
from dotenv import load_dotenv
```

In [ ]:
```
# | hide
load_dotenv()
```

Out[ ]:
```
True
```

In [ ]:
```
from nixtlats import TimeGPT
import os
```

You can instantiate the `TimeGPT` class providing your credentials.

In [ ]:
```
timegpt = TimeGPT(token=os.getenv("TIMEGPT_TOKEN"))
```

In [ ]:
```
# | hide
timegpt = TimeGPT()
```

You can test the validate of your token calling the `validate_token` method:

In [ ]:
```
timegpt.validate_token()
```

```
INFO:nixtlats.timegpt:Happy Forecasting! :), If you have questions or need support,
please email ops@nixtla.io
```

Out[ ]:
```
True
```

Now you can start making forecasts! Let's import an example on the classic `AirPassengers` dataset. This dataset contains the monthly number of airline passengers in Australia between 1949 and 1960. First, let's load the dataset and plot it:

In [ ]:
```python
import pandas as pd
```

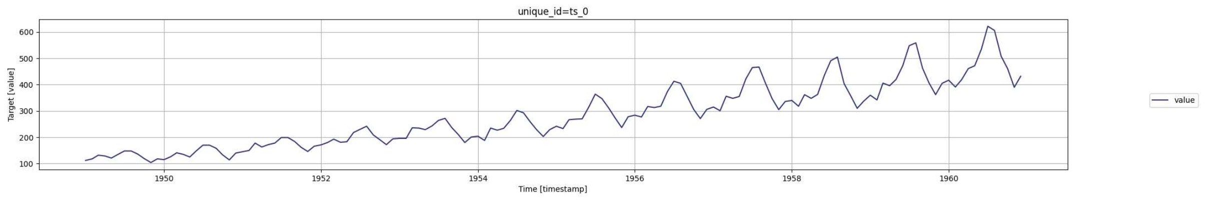
In [ ]:
```python
df = pd.read_csv(
    "https://raw.githubusercontent.com/Nixtla/transfer-learning-time-series/main/da
)
df.head()
```

Out[ ]:

	timestamp	value
0	1949-01-01	112
1	1949-02-01	118
2	1949-03-01	132
3	1949-04-01	129
4	1949-05-01	121

In [ ]:
```python
timegpt.plot(df, time_col="timestamp", target_col="value")
```

Out[ ]:



Important requirements of the data

- Make sure the target variable column does not have missing or non-numeric values.
- Do not include gaps/jumps in the datestamps (for the given frequency) between the first and late datestamps. The forecast function will not impute missing dates.
- The format of the datestamp column should be readable by Pandas (see this link for more details).

Next, forecast the next 12 months using the SDK `forecast` method. Set the following parameters:

- `df` : A pandas dataframe containing the time series data.
- `h` : The number of steps ahead to forecast.
- `freq` : The frequency of the time series in Pandas format. See pandas' available frequencies.
- `time_col` : Column that identifies the datestamp column.
- `target_col` : The variable that we want to forecast.

In [ ]:
```
timegpt_fcst_df = timegpt.forecast(
    df=df, h=12, freq="MS", time_col="timestamp", target_col="value"
)
timegpt_fcst_df.head()
```

```
INFO:nixtlats.timegpt:Validating inputs...
INFO:nixtlats.timegpt:Preprocessing dataframes...
INFO:nixtlats.timegpt:Calling Forecast Endpoint...
```

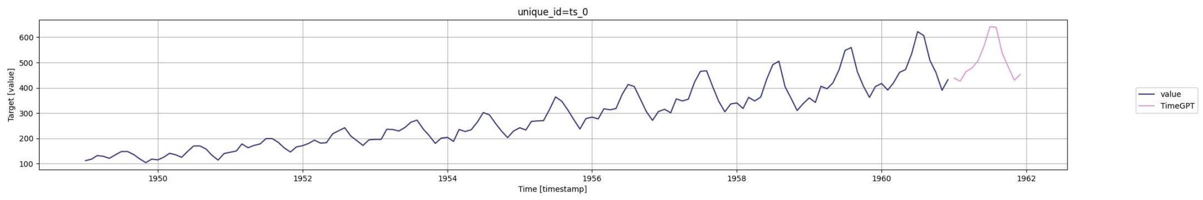
Out[ ]:

	timestamp	TimeGPT
0	1961-01-01	437.837921
1	1961-02-01	426.062714
2	1961-03-01	463.116547
3	1961-04-01	478.244507
4	1961-05-01	505.646484

> BEFORE: *2 API Calls | 146400 Tokens | 245.28 Spent*
> AFTER: *3 API Calls | 146556 Tokens | 245.49 Spent*
> **USAGE: 1 API Call | 156 Tokens | 0.21 Spent**

In [ ]:
```
timegpt.plot(df, timegpt_fcst_df, time_col="timestamp", target_col="value")
```

Out[ ]:



You can also produce a longer forecasts increasing the horizon parameter. For example, let's forecast the next 36 months:

In [ ]:
```
timegpt_fcst_df = timegpt.forecast(
    df=df, h=36, time_col="timestamp", target_col="value", freq="MS"
)
timegpt_fcst_df.head()
```

```
INFO:nixtlats.timegpt:Validating inputs...
INFO:nixtlats.timegpt:Preprocessing dataframes...
WARNING:nixtlats.timegpt:The specified horizon "h" exceeds the model horizon. This m
ay lead to less accurate forecasts. Please consider using a smaller horizon.
INFO:nixtlats.timegpt:Calling Forecast Endpoint...
```

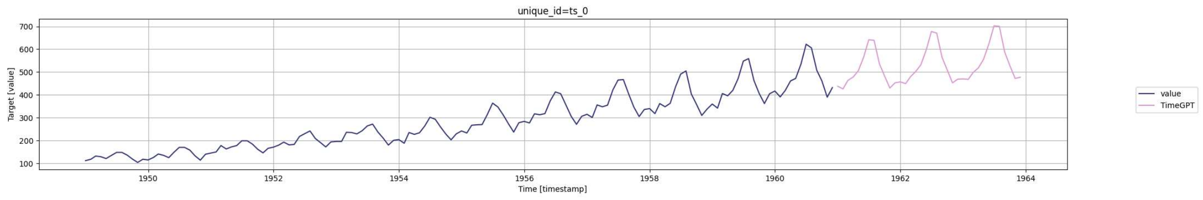
Out[ ]:

	timestamp	TimeGPT
0	1961-01-01	437.837921
1	1961-02-01	426.062714
2	1961-03-01	463.116547
3	1961-04-01	478.244507
4	1961-05-01	505.646484

> BEFORE: *3 API Calls | 146556 Tokens | 245.49 Spent*
> AFTER: *4 API Calls | 146736 Tokens | 245.78 Spent*
> **USAGE: 1 API Call | 180 Tokens | 0.29 Spent**

In [ ]:
```
timegpt.plot(df, timegpt_fcst_df, time_col="timestamp", target_col="value")
```

Out[ ]:

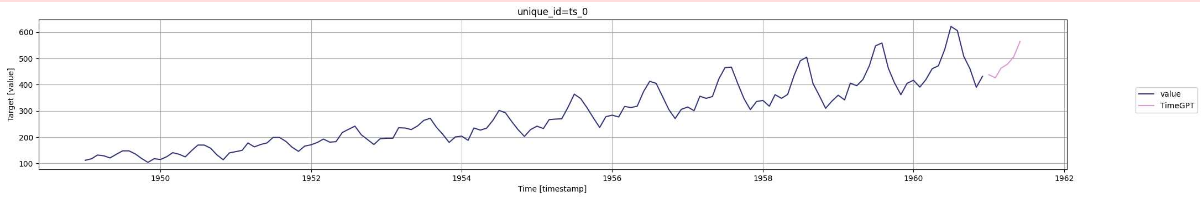


Or a shorter one:

In [ ]:
```
timegpt_fcst_df = timegpt.forecast(
    df=df, h=6, time_col="timestamp", target_col="value", freq="MS"
)
timegpt.plot(df, timegpt_fcst_df, time_col="timestamp", target_col="value")
```

```
INFO:nixtlats.timegpt:Validating inputs...
INFO:nixtlats.timegpt:Preprocessing dataframes...
INFO:nixtlats.timegpt:Calling Forecast Endpoint...
```

Out[ ]:



> BEFORE: *4 API Calls | 146736 Tokens | 245.78 Spent*
> AFTER: *5 API Calls | 146886 Tokens | 245.97 Spent*
> **USAGE: 1 API Call | 150 Tokens | 0.19 Spent**

TimeGPT-1 is currently optimized for short horizon forecasting. While the `forecast` method will allow any positive and large horizon, the accuracy of the forecasts might degrade. We are currently working to improve the accuracy on longer forecasts.

Using DateTime index to infer frequency

The freq parameter, which indicates the time unit between consecutive data points, is particularly critical. Fortunately, you can pass a DataFrame with a DateTime index to the forecasting method, ensuring that your time series data is equipped with necessary temporal features. By assigning a suitable freq parameter to the DateTime index of a DataFrame, you inform the model about the consistent interval between observations — be it days ('D'), months ('M'), or another suitable frequency.

In [ ]:

```
df_time_index = df.set_index("timestamp")
df_time_index.index = pd.DatetimeIndex(df_time_index.index, freq="MS")
timegpt.forecast(df=df, h=36, time_col="timestamp", target_col="value").head()
```

```
INFO:nixtlats.timegpt:Validating inputs...
INFO:nixtlats.timegpt:Preprocessing dataframes...
INFO:nixtlats.timegpt:Inferred freq: MS
WARNING:nixtlats.timegpt:The specified horizon "h" exceeds the model horizon. This m
ay lead to less accurate forecasts. Please consider using a smaller horizon.
INFO:nixtlats.timegpt:Calling Forecast Endpoint...
```

Out[ ]:

	timestamp	TimeGPT
0	1961-01-01	437.837921
1	1961-02-01	426.062714
2	1961-03-01	463.116547
3	1961-04-01	478.244507
4	1961-05-01	505.646484

> BEFORE: *5 API Calls* | *146886 Tokens* | *245.97 Spent*
> AFTER: *6 API Calls* | *147066 Tokens* | *246.26 Spent*
> **USAGE: 1 API Call | 180 Tokens | 0.29 Spent**