8.2 Exercise: Time Series Modeling Assignment by Phil Han

In [27]:
```
# Load necessary packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels import model_selection
from sklearn.linear_model  import LogisticRegression
from sklearn.model_selection import train_test_split
from neuralprophet import NeuralProphet
```

In [28]:
```
# Read the dataset
df = pd.read_csv('us_retail_sales.csv')
df.tail(5)
```

Out[28]:

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
25	2017	416081	415503	414620	416889	414540	416505	416764.0	417179.0	420510.0	426933.0	431158.0	433282.0
26	2018	432148	434106	433232	435610	439996	438191	440703.0	439278.0	438985.0	444038.0	445242.0	434803.0
27	2019	440701	439096	441767	448709	449552	450927	454612.0	456500.0	452843.0	455486.0	457609.0	459855.0
28	2020	460566	459610	434281	379892	444631	476343	485672.0	483716.0	493327.0	493991.0	488652.0	484782.0
29	2021	520162	504458	559871	562269	568987	550782	NaN	NaN	NaN	NaN	NaN	NaN

In [29]:
```
# Explore dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 13 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   YEAR    30 non-null     int64
 1   JAN     30 non-null     int64
 2   FEB     30 non-null     int64
 3   MAR     30 non-null     int64
 4   APR     30 non-null     int64
 5   MAY     30 non-null     int64
 6   JUN     30 non-null     int64
 7   JUL     29 non-null     float64
 8   AUG     29 non-null     float64
 9   SEP     29 non-null     float64
 10  OCT     29 non-null     float64
 11  NOV     29 non-null     float64
 12  DEC     29 non-null     float64
dtypes: float64(6), int64(7)
memory usage: 2.2 KB
```

In [30]:
```
# Check the shape
df.shape
```

Out[30]:
```
(30, 13)
```

In [31]:
```
# Get the description of the dataset
df.describe()
```

Out[31]:

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV
count	30.000000	30.000000	30.000000	30.000000	30.000000	30.000000	30.000000	29.000000	29.000000	29.000000	29.00000	29.00000
mean	2006.500000	304803.833333	305200.900000	307533.566667	306718.600000	309205.633333	311406.966667	304375.448276	305455.965517	306078.206897	307310.82069	307794.896552
std	8.803408	87687.399232	96682.043053	100002.422696	98207.161171	99541.010078	101057.121778	92471.103673	92504.808195	93008.417392	92836.64419	92702.551770
min	1992.000000	147223.000000	146905.000000	146031.000000	149809.000000	149809.000000	150791.000000	150961.000000	152588.000000	152588.000000	153531.00000	153681.000000
25%	1999.250000	228856.750000	231475.750000	233019.000000	233235.500000	234976.500000	235967.250000	233946.000000	236566.000000	237481.000000	237553.00000	240544.000000
50%	2006.500000	303486.000000	304592.500000	308655.500000	311233.500000	308690.000000	312957.000000	313520.000000	310046.000000	310673.000000	310479.00000	308675.000000
75%	2013.750000	379521.000000	377008.500000	379221.000000	379797.500000	382698.250000	383839.750000	373554.000000	372489.000000	372505.000000	373663.50000	373914.000000
max	2021.000000	520162.000000	504458.000000	559871.000000	562269.000000	568987.000000	550782.000000	489272.000000	483716.000000	493327.000000	493991.00000	488652.000000

In [32]:
```
# Reshape the data using Pandas melt method
df =pd.melt(df, id_vars=['YEAR'], var_name = 'Month', value_name = 'Sales')
d = {'JAN':1, 'FEB':2, 'MAR':3, 'APR':4, 'MAY':5, 'JUN':6, 'JUL':7, 'AUG':8, 'SEP':9, 'OCT':10, 'NOV':11, 'DEC':12}
df['Month'] = df['Month'].map(d)
df['Date'] = pd.to_datetime(df['Month', 'YEAR'].astype(Day=1))
df = df.sort_values(by='Date').drop(columns=['Month', 'YEAR']).reset_index(drop=['index'])
```

In [33]:
```
# Check the dataframe for the missing values
df
```

Out[33]:

	Sales	Date
0	146925.0	1992-01-01
1	147223.0	1992-02-01
2	146805.0	1992-03-01
3	148031.0	1992-04-01
4	149809.0	1992-05-01
...	**...**	**...**
355	NaN	2021-08-01
356	NaN	2021-09-01
357	NaN	2021-10-01
358	NaN	2021-11-01
359	NaN	2021-12-01

360 rows × 2 columns

In [34]:
```
# Impute data for missing values
#df['Sales'] =df['Sales'].fillna(df['Sales'].mean())
df['Sales'] =df['Sales'].fillna(method='ffill')
```

In [35]:
```
# Check the dataframe after handling the missing values
df
```

Out[35]:

	Sales	Date
0	146925.0	1992-01-01
1	147223.0	1992-02-01
2	146805.0	1992-03-01
3	148031.0	1992-04-01
4	149809.0	1992-05-01
...	**...**	**...**
355	550782.0	2021-08-01
356	550782.0	2021-09-01
357	550782.0	2021-10-01
358	550782.0	2021-11-01
359	550782.0	2021-12-01

360 rows × 2 columns

In [36]:
```
# Rename the dataframe with date and sales as 'ds' and 'y', respectively
df = df[['Date', 'Sales']]
df.rename(columns={"Date": "ds", "Sales":"y"}, inplace=True)
```

In [37]:
```
# 1. Plot the data for sales over time
ax = df.set_index('ds').plot(figsize=(15, 12))
ax.set_ylabel('US Retail Sales')
ax.set_xlabel('Date')
ax.legend(['Sales'])

plt.title('U.S. Retail Sales Over the Years')
plt.show()
```

The above graph shows that the retailers suffered a significant drop in sales during the 2008-2009 recession. But during the beginning of the pandemic in 2020, there was even a significant decline in retail sales as indicated by a cliff. Once the retail hit its rock bottom, it slowly rebounded up and climbed past the pre-pandemic high.

In [38]:
```
# 2. Split the data into a training and test set. Use the last year of data (July 2020 - June 2021) of data as
# the test set and the remaining as the training set.
df_train = df[df['ds'] < '2020-07-01']
df_test = df[df['ds'] >= '2020-07-01']
```

In [39]:
```
df_test.tail(10)
```

Out[39]:

	ds	y
350	2021-03-01	550870.0
351	2021-04-01	562269.0
352	2021-05-01	568987.0
353	2021-06-01	550782.0
354	2021-07-01	550782.0
355	2021-08-01	550782.0
356	2021-09-01	550782.0
357	2021-10-01	550782.0
358	2021-11-01	550782.0
359	2021-12-01	550782.0

In [40]:
```
# 3. Use the training set to build a predictive model for the monthly retail sales
m = NeuralProphet()  # Initialize the predictive model with Neural Prophet, using default variables

# Fit the train data and the test data from #2 to the predictive model built w/ Neural Prophet
metrics_train = m.fit(df=df_train, freq="M")
metrics_test = m.test(df=df_test)
```

```
INFO - (NP.utils_torch.lr_range_test) - Max frequency MO corresponds to 91.228% of the data.
INFO - (NP.df_utils._infer_frequency) - Defined frequency is equal to major frequency - M
INFO - (NP.config.init_data_params) - Setting normalization to global as only one dataframe provided for training.
INFO - (NP.utils.set_auto_seasonality) - Disabling weekly seasonality. Run NeuralProphet with weekly_seasonality=True to override this.
INFO - (NP.utils.set_auto_seasonality) - Disabling daily seasonality. Run NeuralProphet with daily_seasonality=True to override this.
INFO - (NP.config.set_auto_batch_epoch) - Auto-set batch_size to 16
INFO - (NP.config.set_auto_batch_epoch) - Auto-set epochs to 286
Epoch[286/286]: 100%|▉▉▉
```