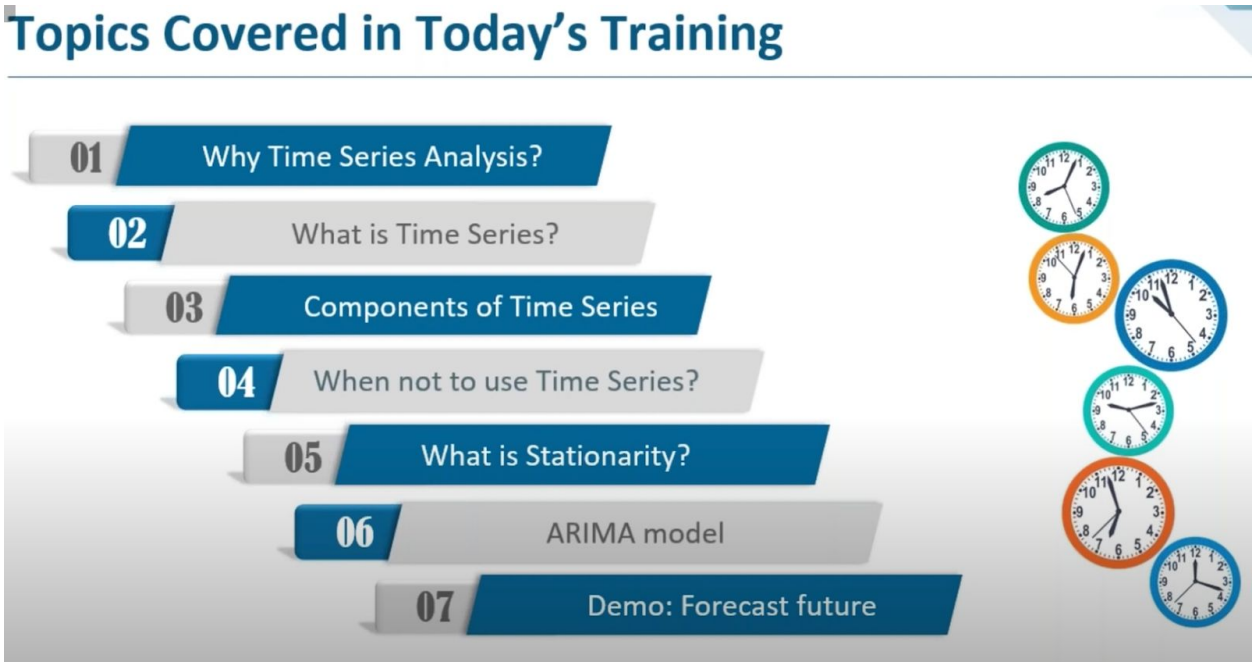
Time Series Analysis in Python | Time Series Forecasting | Data Science with Python | Edureka

eureka! channel

https://www.youtube.com/watch?v=e8Yw4alG16Q



Why Time Series Analysis?

In this analysis, you just have one variable – **TIME**

You can analyse this time series data in order to extract meaningful statistics and other characteristics

What Is Time Series?

- A time series is a set of observation taken at specified times usually at equal intervals
- It is used to predict the future values based on the previous observed values

- Business Forecasting
- Understand past behavior
- Plan future
- Evaluate current accomplishment

Components Of Time Series

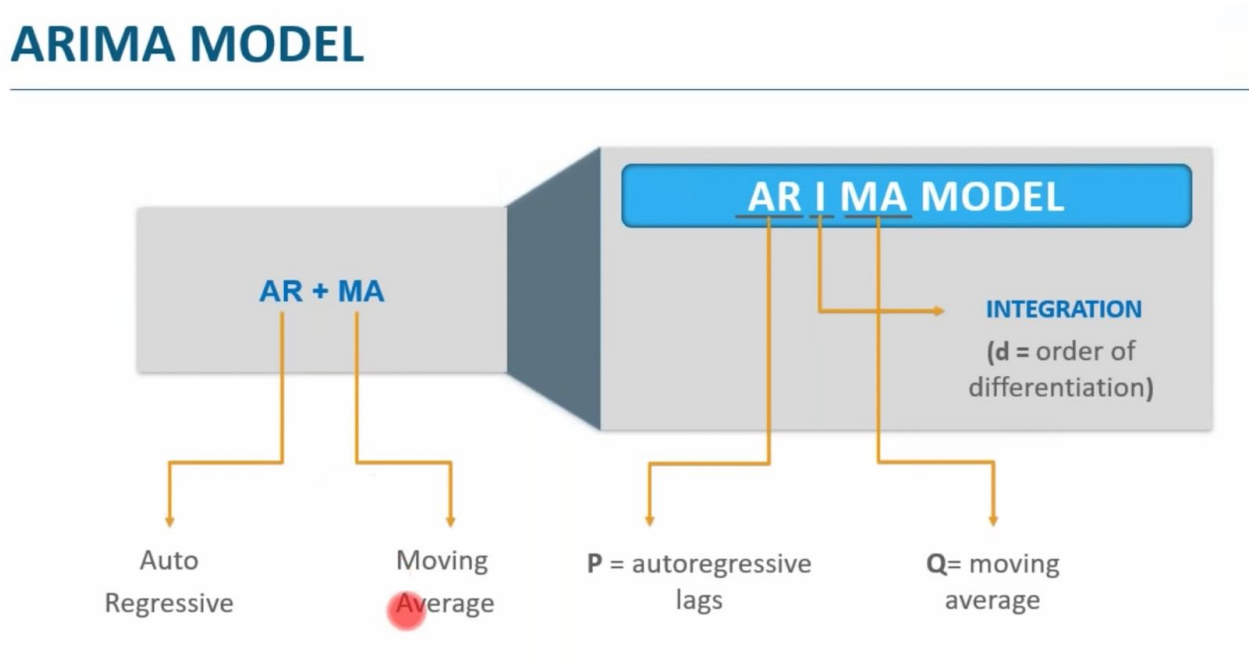
What Is Stationarity?

TS has a particular behaviour over time, there is a very high probability that it will follow the same in the future.

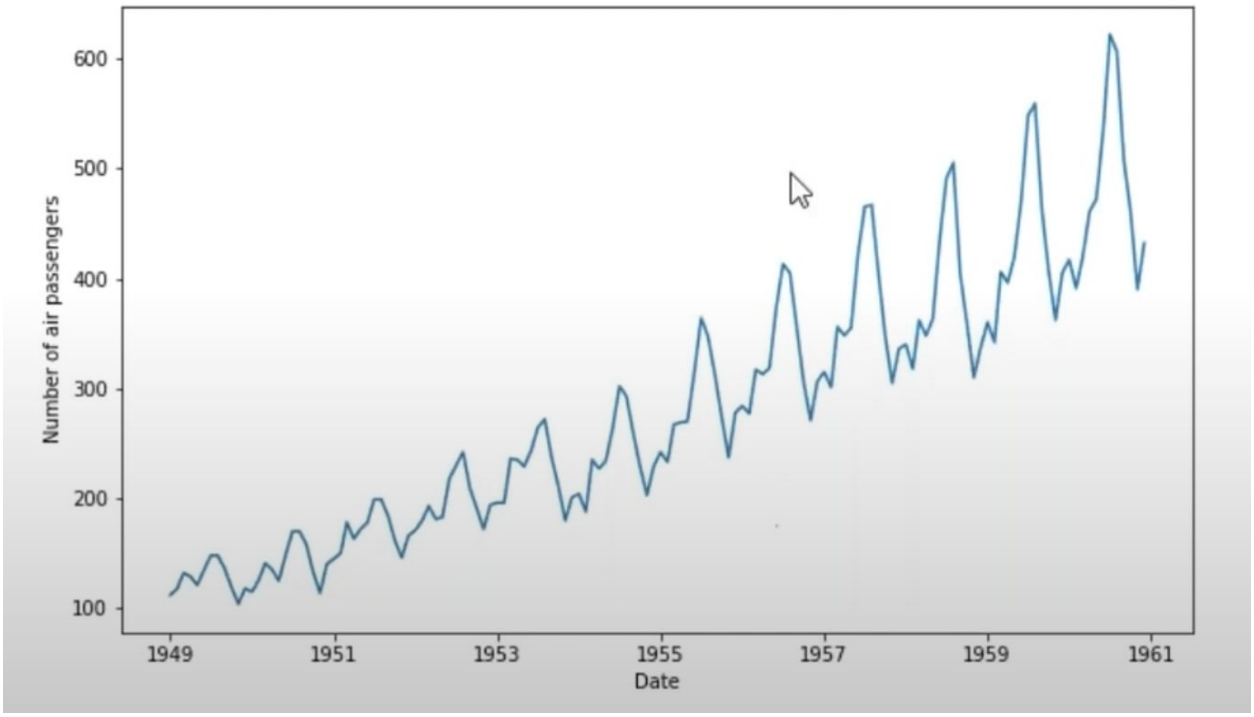
How to remove Stationarity?

1. Constant mean
2. Constant Variance
3. Autocovariance that does not depend on time

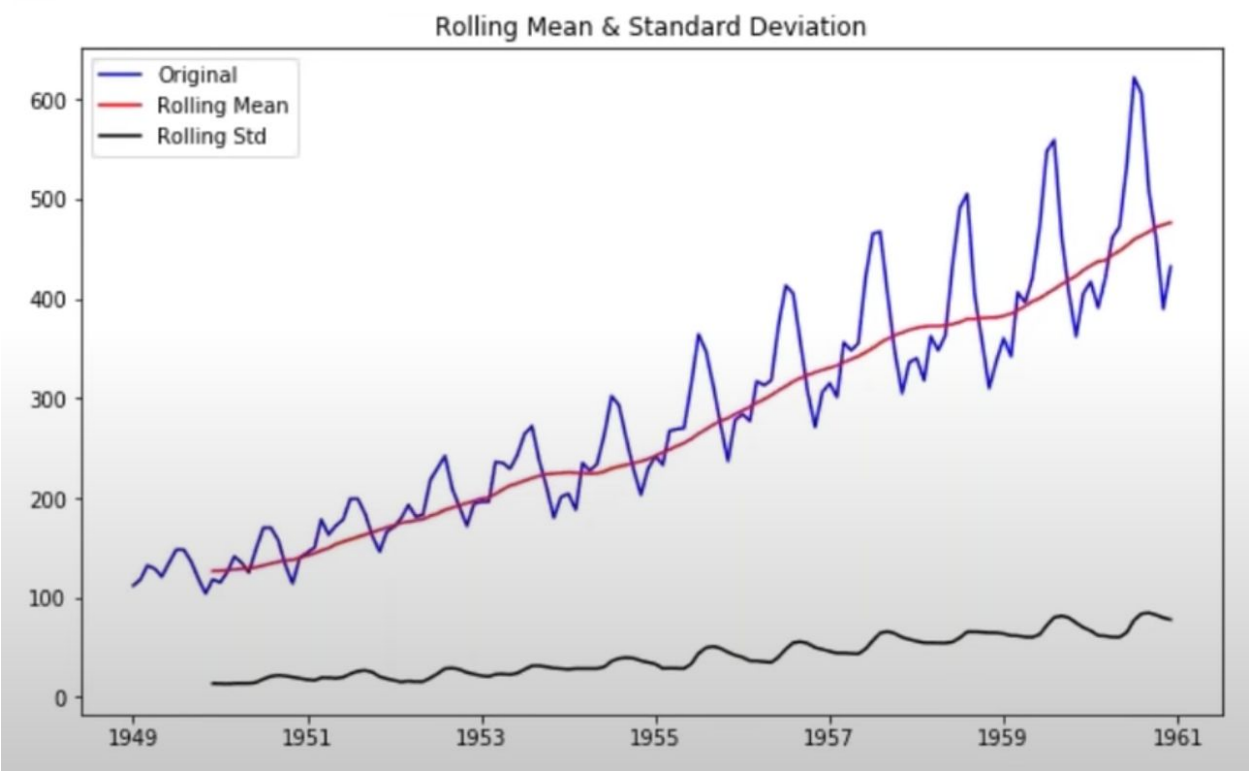
ARIMA MODEL



Not stationary data, conclusion reached by the non constant average (compare 1949 average with 1960 average). Graph below.

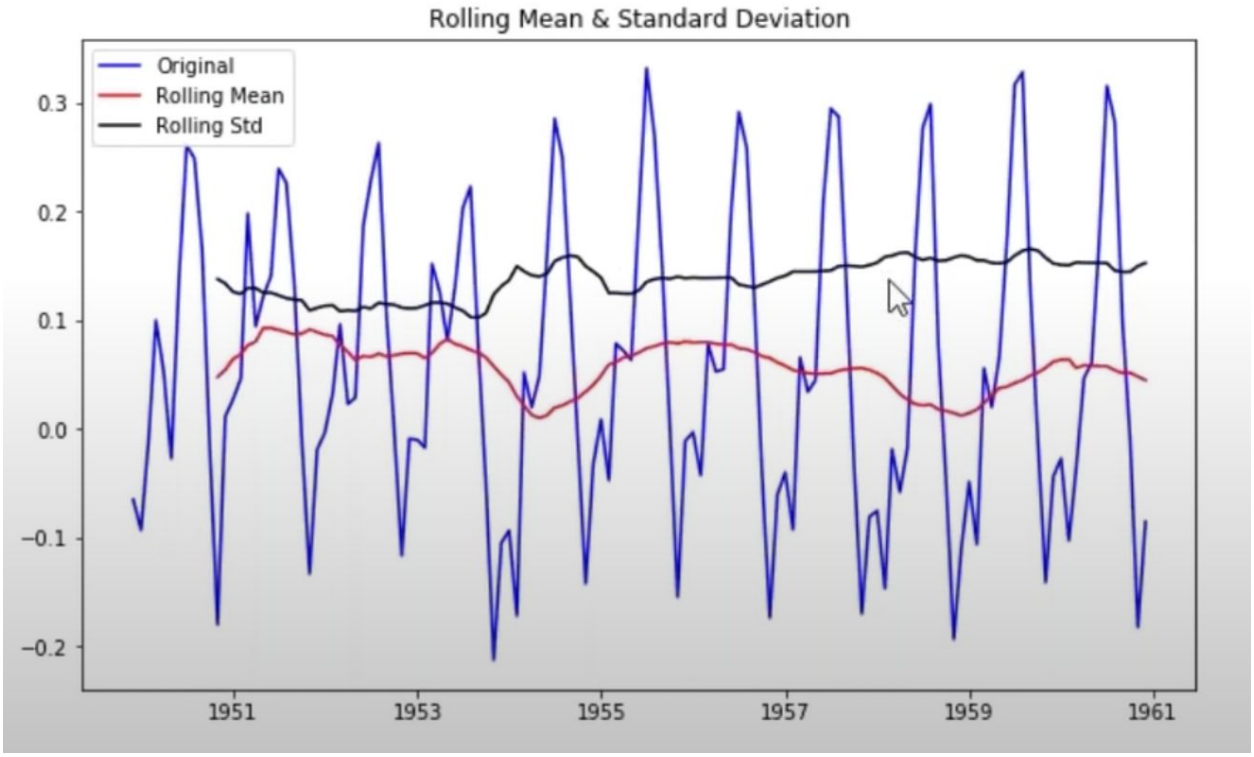


To be even more explicit, use the rolling average and standard deviation metrics.




```
Results of Dickey-Fuller Test:
Test Statistic                   0.815369
p-value                          0.991880
#Lags Used                      13.000000
Number of Observations Used    130.000000
Critical Value (1%)             -3.481682
Critical Value (5%)             -2.884042
Critical Value (10%)            -2.578770
dtype: float64
```

After transformation (log and subtracting the moving average over the log), we have:

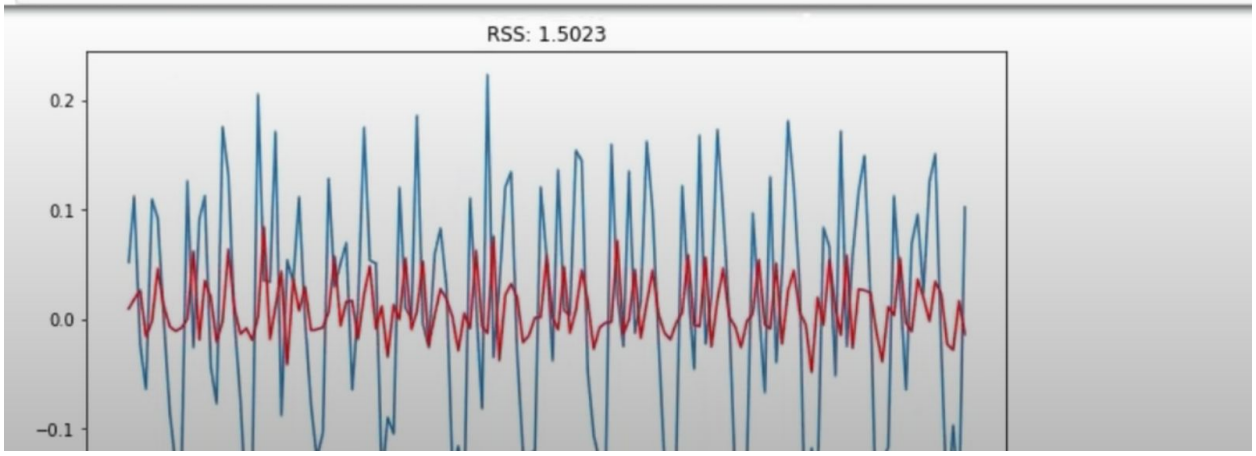



```
Results of Dickey-Fuller Test:
Test Statistic                  -3.162908
p-value                          0.022235
#Lags Used                      13.000000
Number of Observations Used    119.000000
Critical Value (1%)             -3.486535
Critical Value (5%)             -2.886151
Critical Value (10%)            -2.579896
dtype: float64
```

Training of the ARIMA model

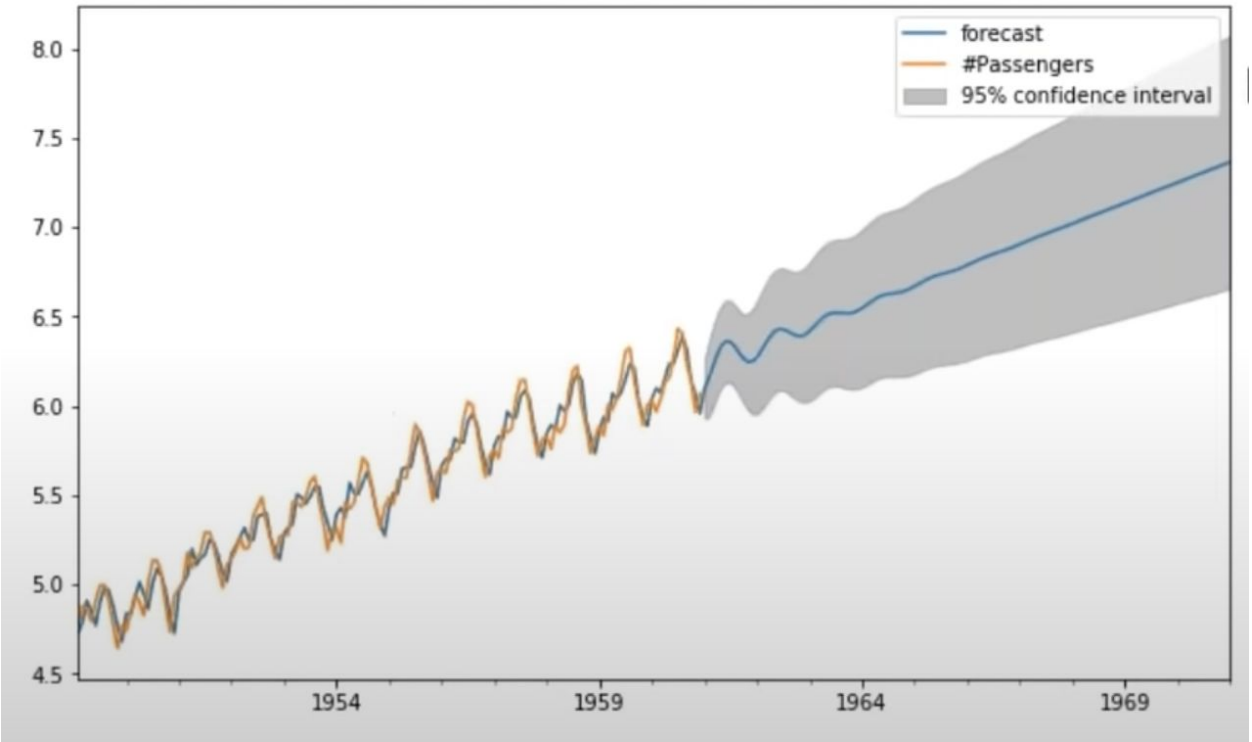
```
from statsmodels.tsa.arima_model import ARIMA

#AR MODEL
model = ARIMA(indexedDataset_logScale, order=(0, 1, 0))
results_AR = model.fit(disp=-1)
plt.plot(datasetLogDiffShifting)
plt.plot(results_AR.fittedvalues, color='red')
plt.title('RSS: %.4f'% sum((results_AR.fittedvalues-datasetLogDiffShifting["#Passengers"])**2))
print('Plotting AR model')
```



Then prediction for the next 10 years (120 months).

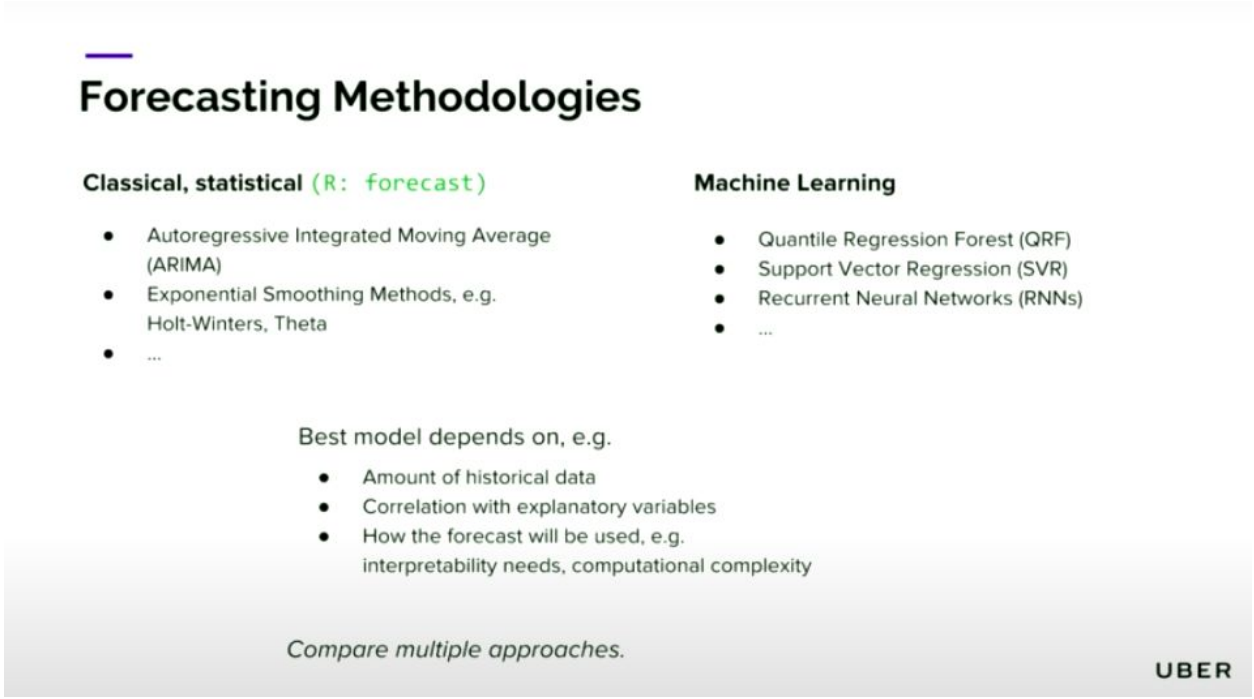
```
results_ARIMA.plot_predict(1,264)
x=results_ARIMA.forecast(steps=120)
```


forecast
#Passengers
95% confidence interval
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
1954
1959
1964
1969

Introduction to Forecasting in Machine Learning and Deep Learning

InfoQ channel

https://www.youtube.com/watch?v=bn8rVBuIcFg



Evaluation: Backtesting

Chronological testing

Sliding Window

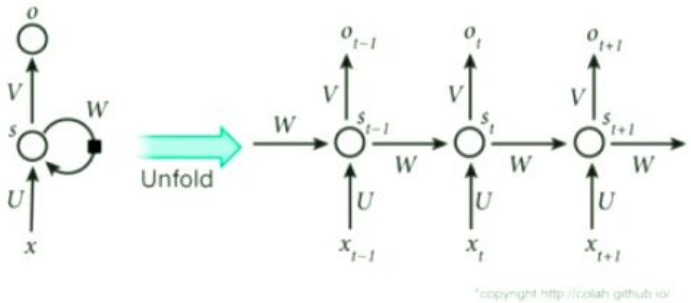
Figure 1: In the sliding window backtesting model, a fixed-size training window (in black) slides over the entire history of a time series and is repeatedly tested against a forecasting window (in orange) with older data points dropped.

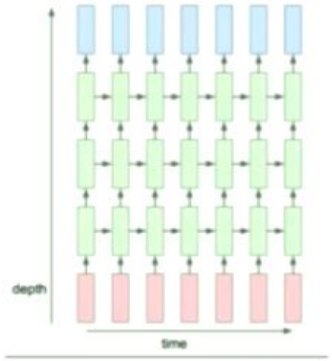
Expanding Window

Figure 2: In the expanding window backtesting model, a training window (in black) expands over the entire history of a time series and is repeatedly tested against forecasting window (in orange) without dropping older data points.

Evaluation Metric: Compare to Naive Forecast

Recurrent Neural Networks (RNNs)





RNNs can be multi-layer

Advantages

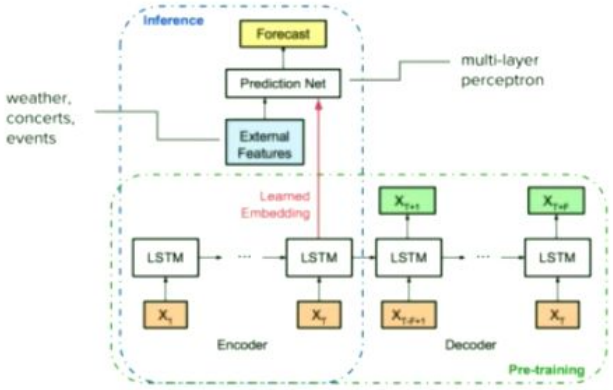
- end-to-end modeling,
- ease of incorporating exogenous variables,
- captures non-linear feature interactions
- automatic feature extraction.

Challenges

- need sufficient data for training,
- interpretability,
- need uncertainty estimates.

Forecasting Time Series for Extreme Events

LSTM Encoder-Decoder + Prediction Network



- **Encoder-Decoder**: two-layer LSTM cells, with 128 and 32 hidden states, respectively.
- **Prediction network**: three fully connected layers with tanh activation, with 128, 64, and 16 hidden units, respectively.
- **Raw data**: log-transformed to alleviate exponential effects.