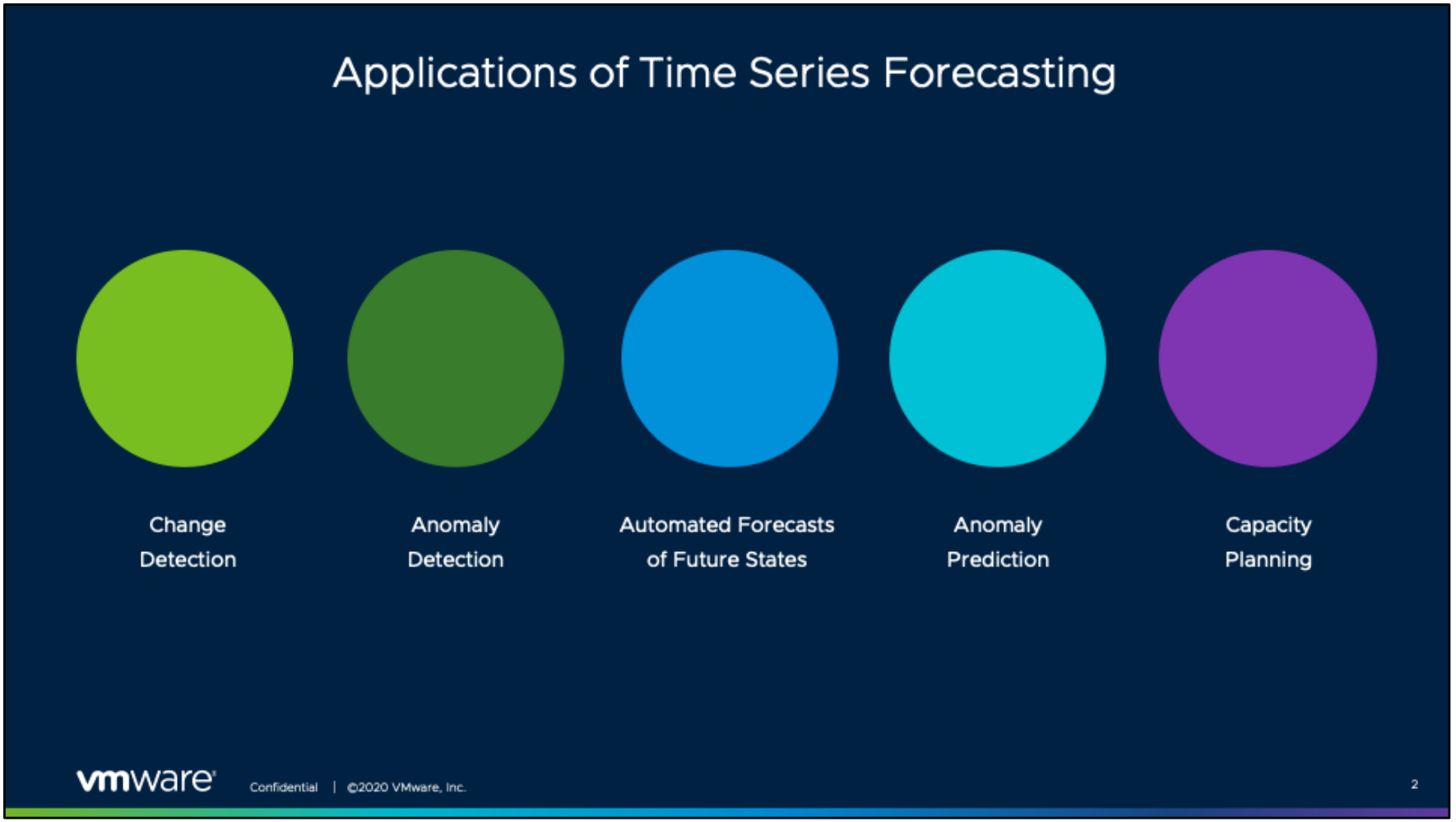
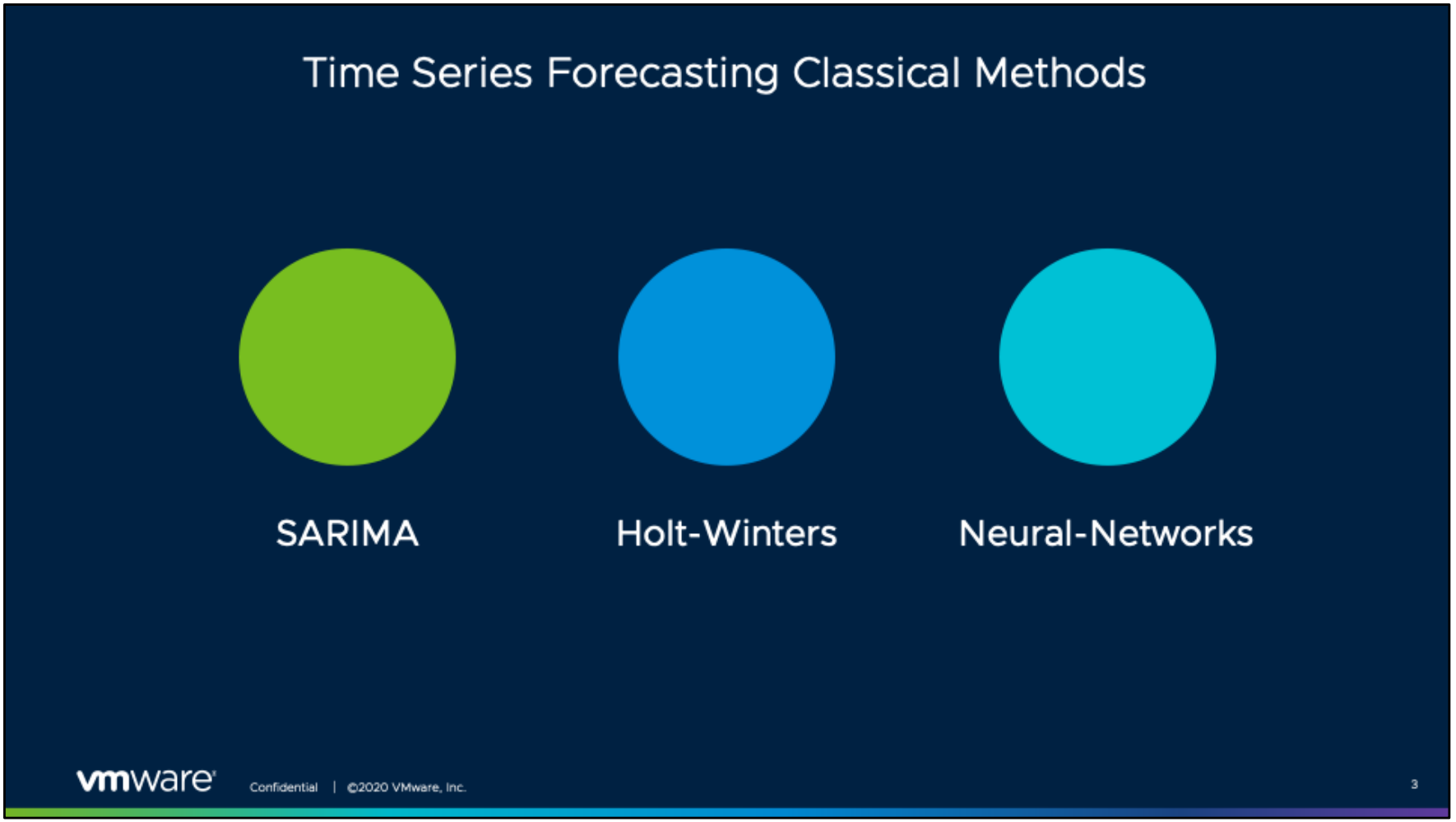
- One of the key components for application performance monitoring/management (APM) software is the availability of Analytics with artificial intelligence for IT operations known as AIOps.
- Time series data (known also as metrics), together with logs, traces, histograms and events is an intrinsic APM-acquired data type heavily utilized by all APM leaders.
- We suggest NN-based time series forecasting system (named as W-TSF) that essentially differs from other well-known classical techniques.
- Implementation and testing were performed in Wavefront by VMware.
- Wavefront offers real-time metrics monitoring and analytics platform designed for optimization of cloud and modern applications that rely on containers and microservices.



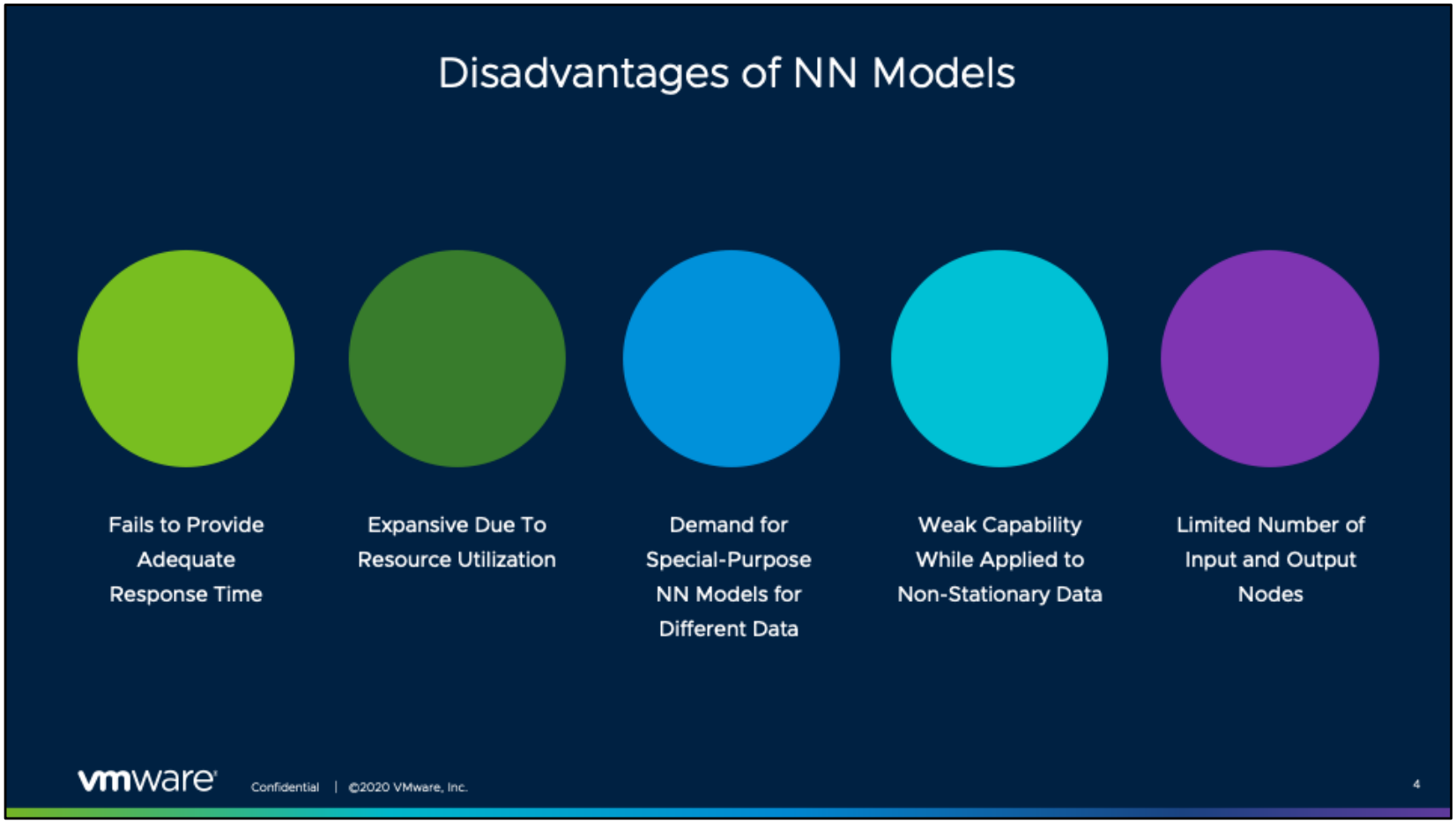
- Fast and accurate time series analysis and forecasting is of great importance for various reasons like anomaly detection, anomaly prediction and capacity planning.
- When metric data are collected and analyzed by a monitoring system, administrators of a cloud environment may desire automated forecasts of future metric-data values indicative of likely future states of applications or infrastructure components.
- Data related to computing-resources and capacities may include trends indicating that additional processor bandwidth or mass-storage capacity may be needed due to increasing workloads, in order to prevent delays and failures and/or to maximize economic efficiency.
- Time series data can also be used for correlation analysis and as a source of anomaly events for further root cause analysis.



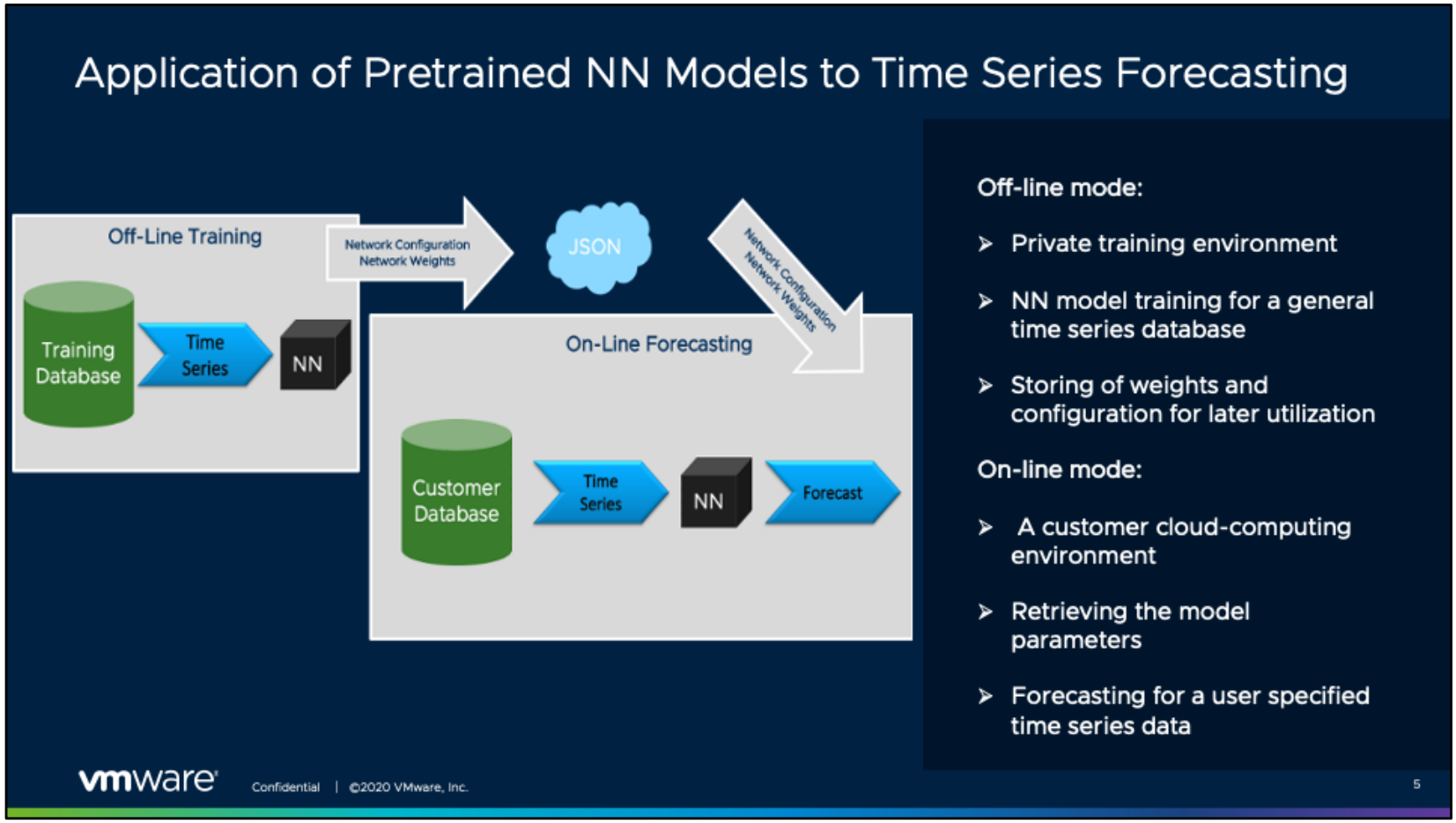
Details see in:

Hyndman, R.J., Athanasopoulos, G.: Forecasting: Principles and Practice, 2018, Monash University, Australia. Available online: https://otexts.com/fpp2/.

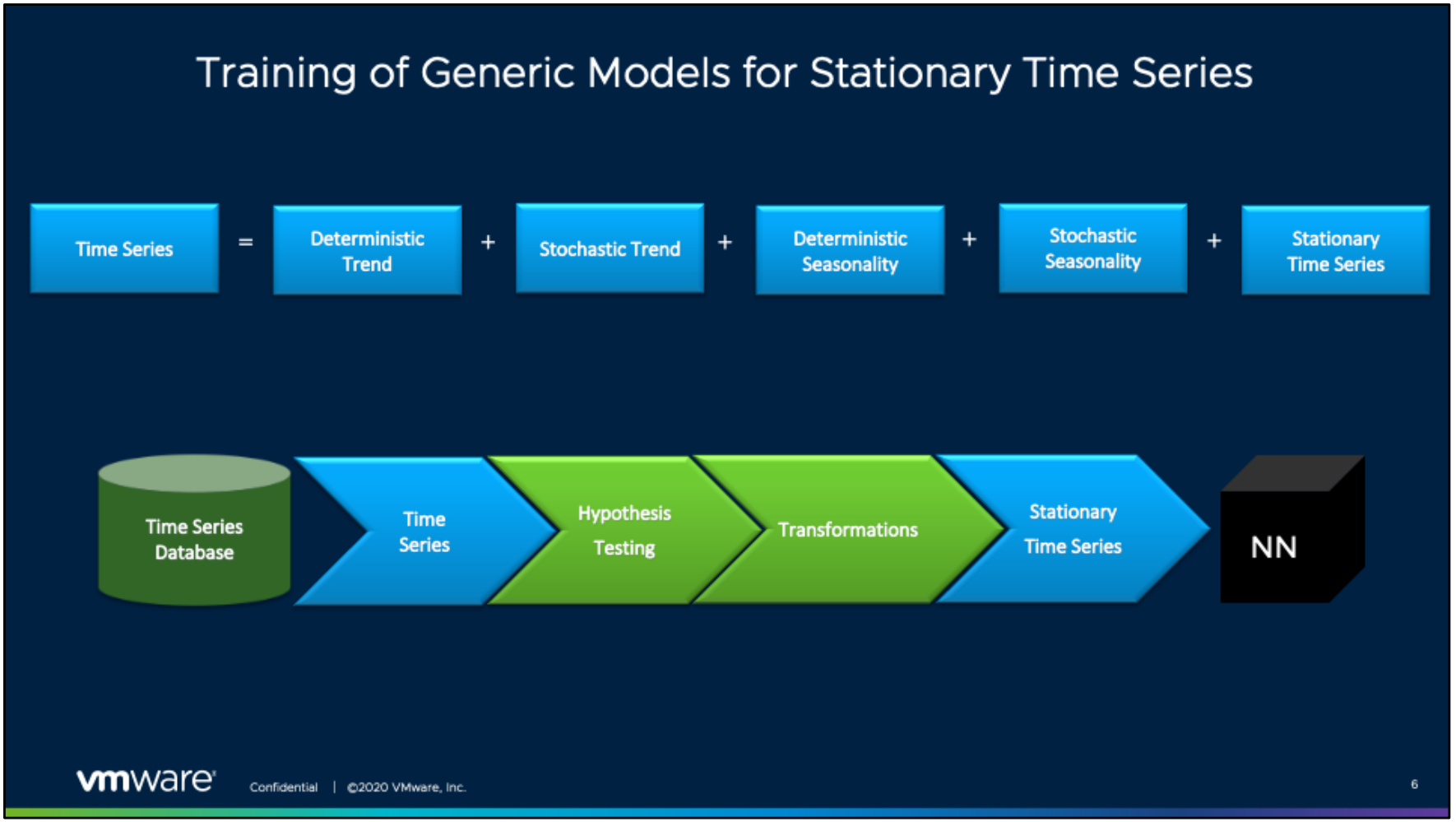
Lewis, N.D.: Deep Time Series Forecasting with Python: An Intuitive Introduction to Deep Learning for Applied Time Series Modeling. CreateSpace Independent Publishing Platform, 2016.



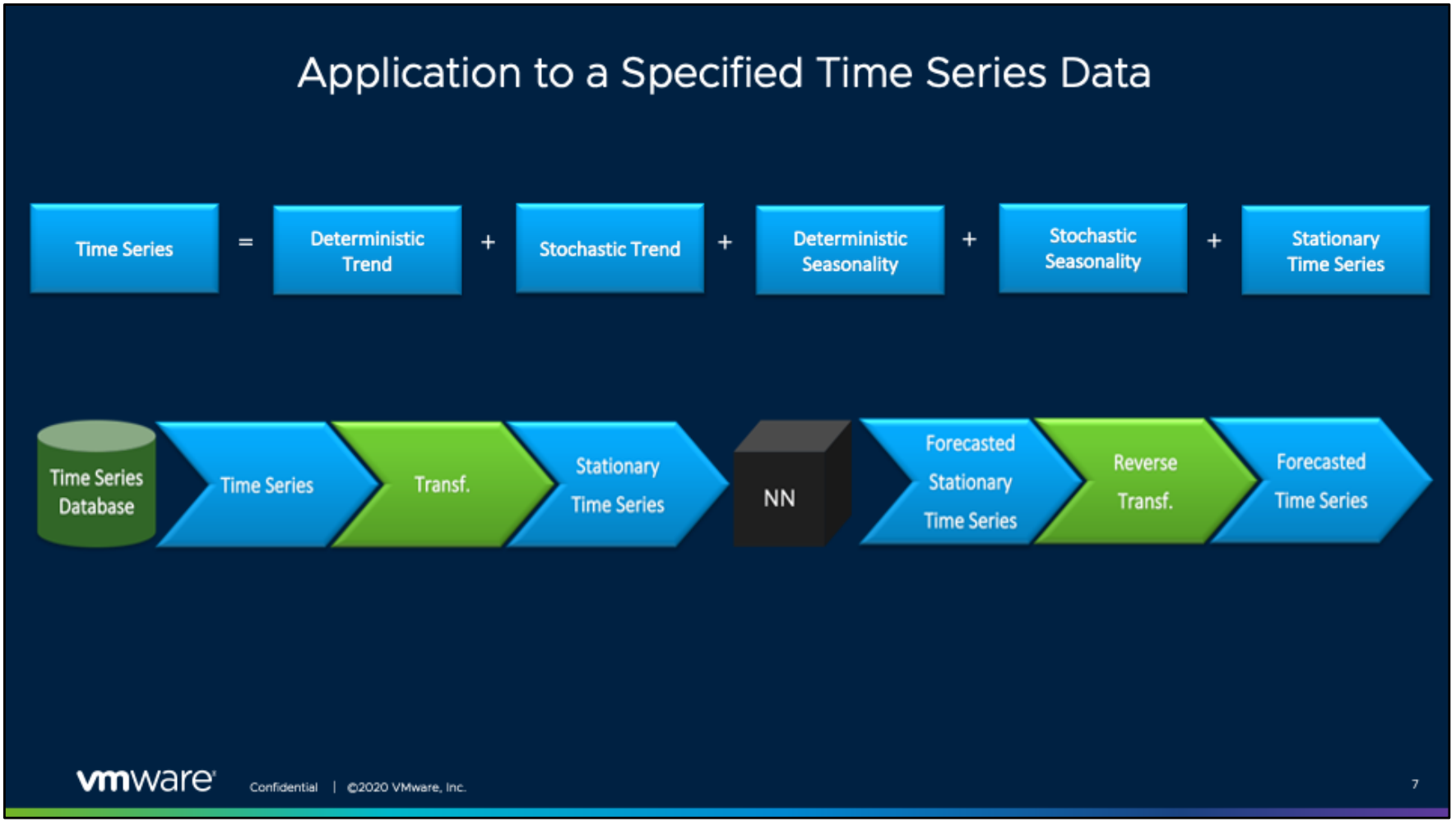
- Application of neural networks (NN) may produce an efficient approach to time-series analysis and forecasting.
- However, naïve implementation of a NN-based system in a cloud-computing environment would likely fail to provide adequate response times and would likely be far too expensive for most clients. Training and storing of neural networks are both time-consuming and expensive with respect to the necessary resources.
- Hence, it is not feasible to train those models in demand for the specified time series data.
- From the other side, it would not be feasible to train and store special-purpose neural networks for all different possible types of time series.
- A naïve attempt to train a single neural network to analyze all of the various different types of time-series data would also likely fail, since different types of time-series data exhibit different types of behaviors and temporal patterns, and because a single neural network would need a vast number of nodes and even vaster sets of training data to produce reasonable forecasts for general time-series data.



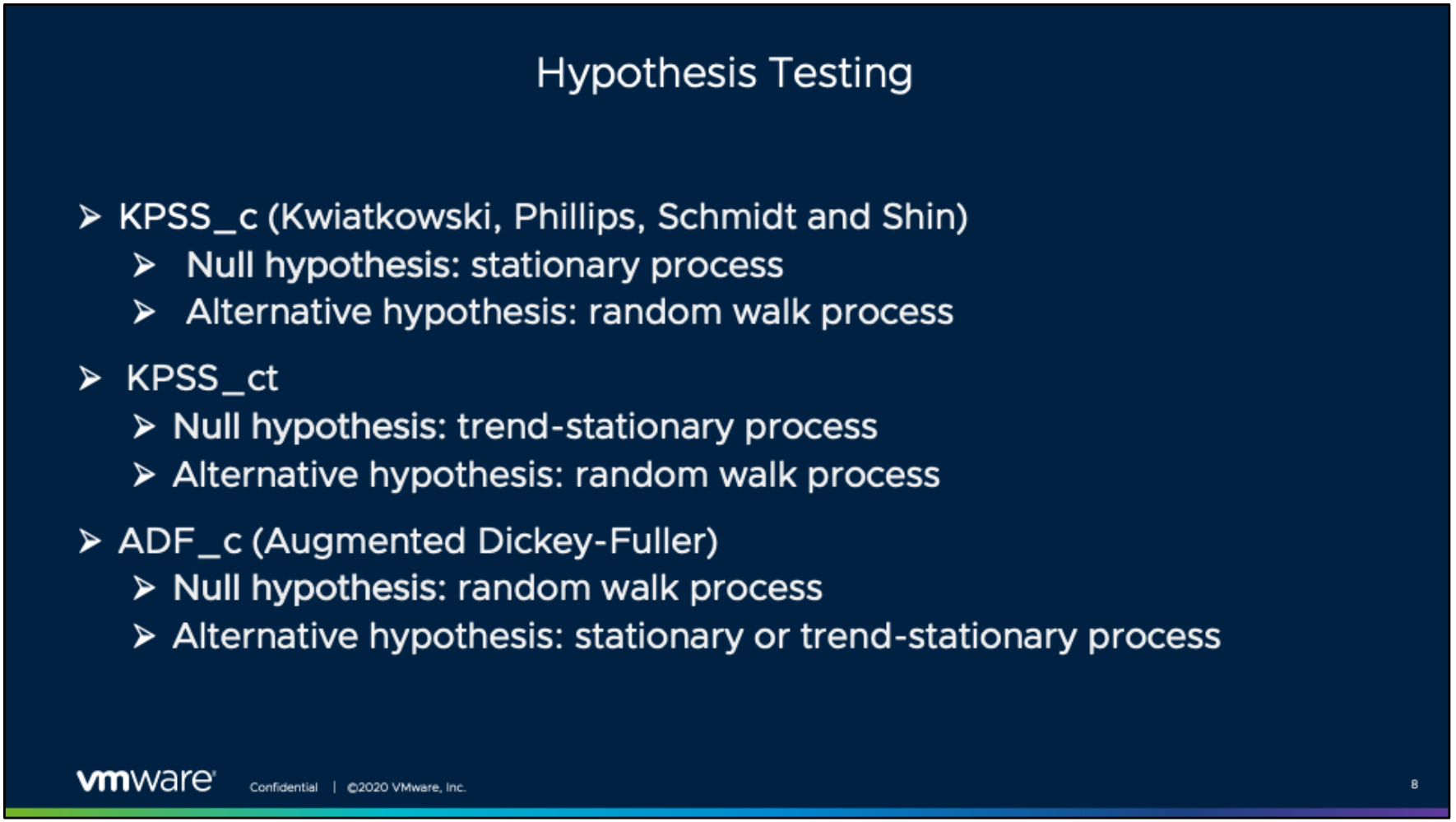
- We follow the common idea of training a general network (pretrained networks) and storing it for further application to forecasting.
- This will help to reduce the resource utilization as training should be performed in private cloud environments where utilization of enough powerful GPUs is possible.
- The slide shows implementation of this approach. The entire system consists of two separated parts: off-line and on-line modes.
- Off-line mode performs model training for a special time series database consisting of time series data across different customers. The weights of the trained network and its configuration is stored in a cloud.
- On-line mode corresponds to a customer cloud-computing environment. The weights and configuration of the pretrained network can be restored from the file and applied to the specified time series data.



- NN models have weak forecasting capabilities while applied to non-stationary time series data.
- The set of stabilizing transformations is time series class specific. Say, the class of time series data with a deterministic trend can be stabilized via detrending by a regression (linear or non-linear), the class with a stochastic trend by a differencing of the proper order, etc.
- Our implementation applies different well-known hypothesis testing algorithms for time series classification.
- Training of NN-models via unknown-type time series needs preliminary classification (categorization) with further transformation to a stationary time series for feeding the network.



- Application of pretrained neural networks to a user specified unknown-type time series needs:
  - Preliminary classification;
  - Transformation to a stationary time series;
  - Application of a pretrained network;
  - Getting the corresponding forecast;
  - Applying reverse transformations for returning to the original scale, trend and seasonality.



Hypothesis testings for time series behavior categorization

KPSS Test

- Data model

$$y_t = c + \lambda t + u_t + e_t$$

c is the level

λt is the deterministic trend

u_t is a stationary process

e_t is a residual

- Test uses Ordinary Least Squares to find the coefficients

Details see in:
Kwiatkowski, D., Phillips, P., Schmidt, P., Shin, Y.: Testing the Null Hypothesis of Stationarity Against the Alternative of a Unit Root: How Sure are We that Economic Time Series have a Unit Root? Journal of Econometrics 54, 159-178 (1992).

KPSS Test

- Partial sum process of residuals

$$S_t = \sum_{i=1}^{t} e_i, \qquad t = 1,2,\dots,T$$

- Lagrange Multiplier (LM) statistic

$$LM = \sum_{i=1}^{T} \frac{S_i^2}{s^2(l)}$$

- $s^2(l)$ is the consistent estimator of σ^2 according to Kwiatkowski et al (1992)
- p-values are calculated by Kwiatkowski et al (1992)
- *Kwiatkowski, D.; Phillips, P. C. B.; Schmidt, P.; Shin, Y. (1992). "Testing the null hypothesis of stationarity against the alternative of a unit root". Journal of Econometrics. 54 (1–3): 159–178.*

Details see in:
Kwiatkowski, D., Phillips, P., Schmidt, P., Shin, Y.: Testing the Null Hypothesis of Stationarity Against the Alternative of a Unit Root: How Sure are We that Economic Time Series have a Unit Root? Journal of Econometrics 54, 159-178 (1992).

ADF Test

- Data model

$$\Delta y_t = y_t - y_{t-1} = c + \varphi y_{t-1} + \sum_{i=1}^{p} \alpha_i \Delta y_{t-i} + u_t + e_t$$

c is the level
u_t is a stationary process
p is the number of lags used in the model
e_t is the residual

- Test uses Ordinary Least Squares to find the coefficients
- Test applies Akaike information criterion for automatic lag selection

Details see in:
Dickey, D.A., Fuller, W.A.: Distribution of the Estimators for Autoregressive Time Series with a Unit Root. J. Amer. Stat. Assoc. 74, 427–431 (1979).
Dickey, D.A., Fuller, W.A.: Likelihood Ratio Statistics for Autoregressive Time Series with a Unit Root. Econometrica 49, 1057-1072 (1981).
Said, E., Dickey, D.A.: Testing for Unit Roots in Autoregressive Moving Average Models of Unknown Order. Biometrika 71, 99–607 (1984).

ADF Test

- The unit root test is then carried out under the null hypothesis $\varphi = 0$ against the alternative hypothesis $\varphi < 1$
- Once a value for the test statistic

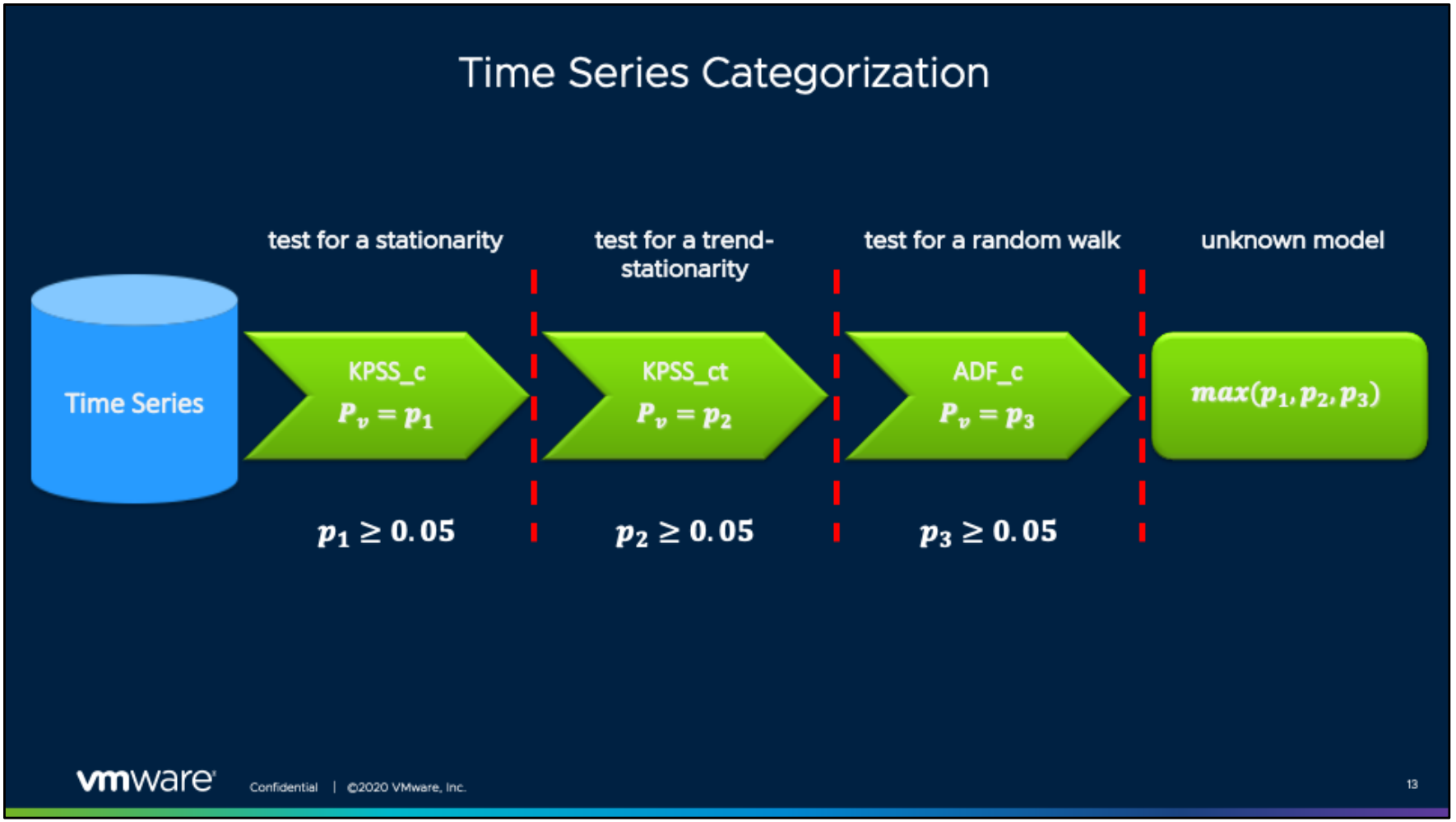
$$DF = \frac{\hat{\varphi}}{SE(\hat{\varphi})}$$

is computed it can be compared to the relevant critical value for the Dickey–Fuller test

- The test is asymmetrical. If the calculated test statistic is less than the critical value, then the null hypothesis of $\varphi = 0$ is rejected, and no unit root is present
- Said, S.E. and D. Dickey (1984). "Testing for Unit Roots in Autoregressive Moving-Average Models with Unknown Order," Biometrika, 71, 599-607

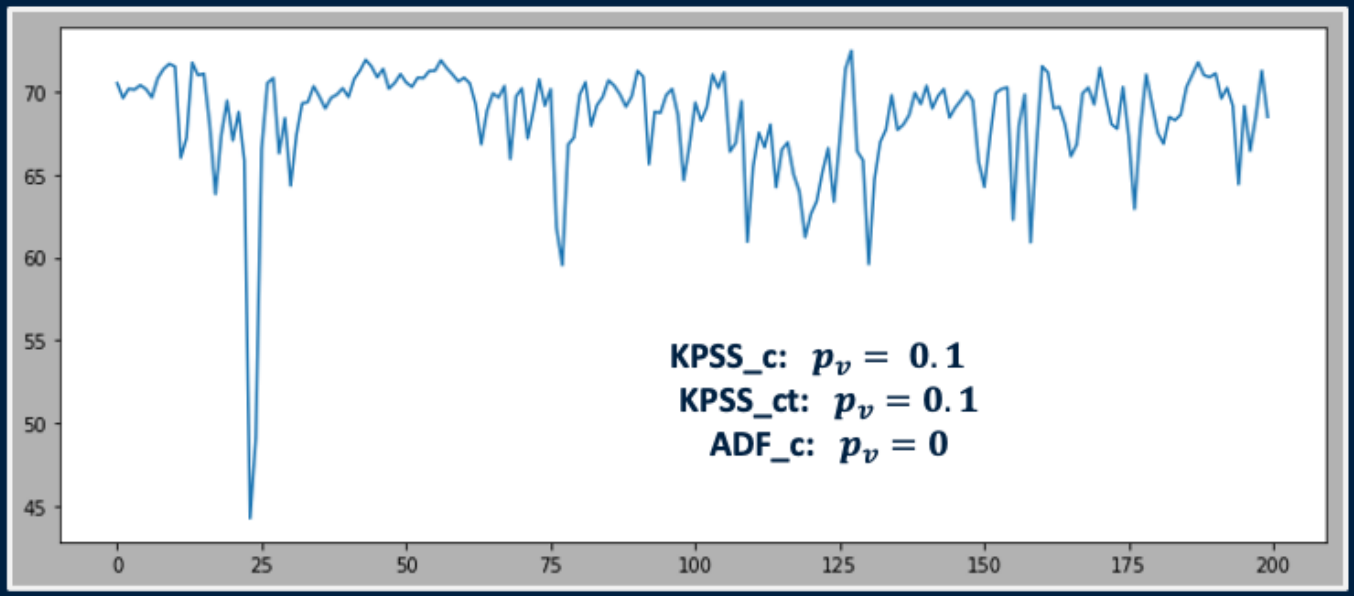
Details see in:

Dickey, D.A., Fuller, W.A.: Distribution of the Estimators for Autoregressive Time Series with a Unit Root. J. Amer. Stat. Assoc. 74, 427–431 (1979).
Dickey, D.A., Fuller, W.A.: Likelihood Ratio Statistics for Autoregressive Time Series with a Unit Root. Econometrica 49, 1057-1072 (1981).
Said, E., Dickey, D.A.: Testing for Unit Roots in Autoregressive Moving Average Models of Unknown Order. Biometrika 71, 99–607 (1984).



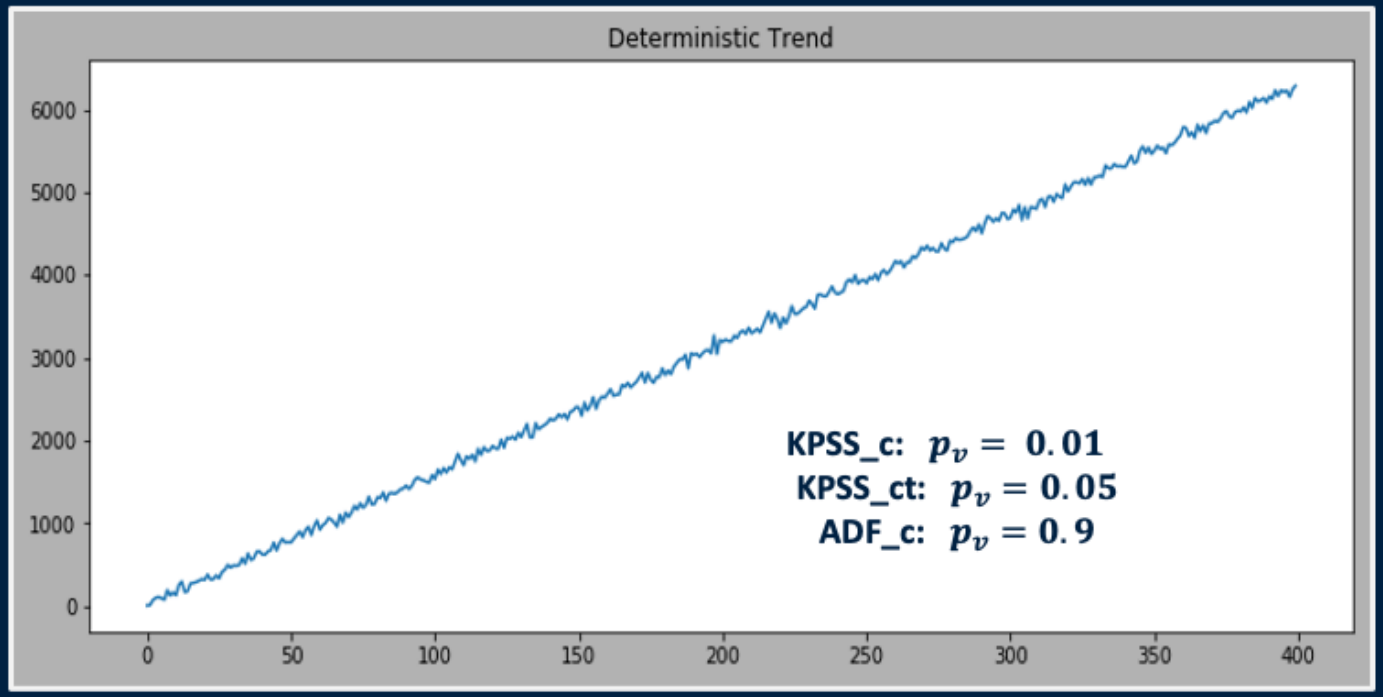
- Chain of hypothesis testings for data categorization.
- We stop the procedure if the p-value of the corresponding test is bigger than 0.05.

Stationary Time Series: Winner is KPSS_c



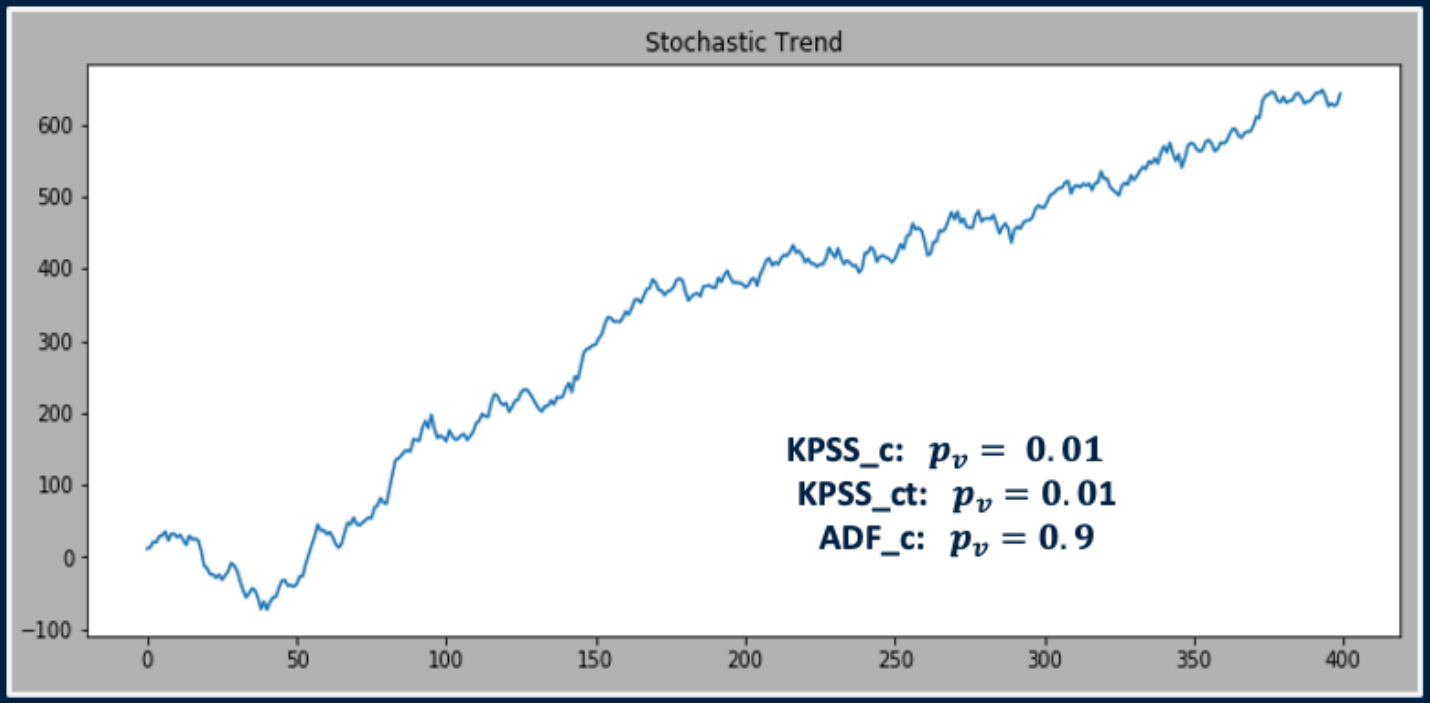
- Hypothesis testing process for a stationary time series.
- The p-value corresponding to the KPSS_c test is bigger than 0.05. It means that the time series is stationary.
- The priority of test applications is important. Although the p-value of the KPSS_ct test is also bigger than 0.05, but the winner is the KPSS_c test which was checked earlier.

Trend-Stationary Time Series: Winner is KPSS_ct



- The p-value of the KPSS_c test is smaller than 0.05. It means that time series is non-stationary.
- The p-value of the next KPSS_ct test is bigger (or equal) than 0.05. We stop the process of categorization and claim the time series as trendy-stationary.
- Although the p-value of the ADF_c is bigger than the p-value of the KPSS_ct test, the winner is the previous one with bigger priority. Also worth noting that the p-values of different tests are uncompareable. We can not state that the time series is probably more a unit-root process rather than a trend-stationary one via comparison of the corresponding p-values.

Unit-Root Process: Winner is ADF_c



- KPSS_c and KPPS_ct have p-values smaller than 0.05
- ADF_c test is the only winner. We see a unit-root process.

Transformations

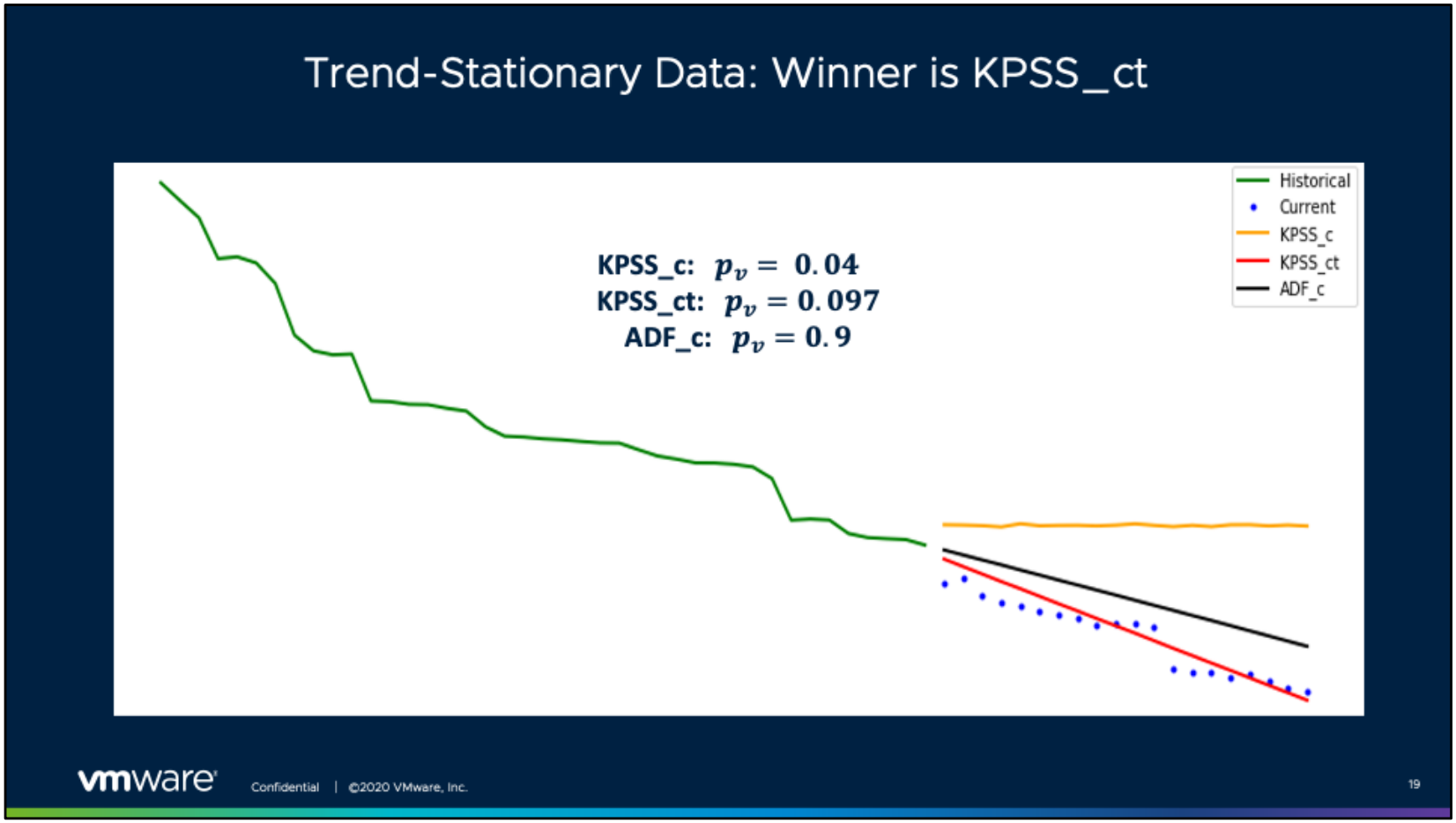
- Stationary process
  - Without transformations
- Trend-stationary process
  - Linear regression for trend elimination
- Random walk (with drift) process
  - Differencing of required order

- Each time series data category has its own set of transformations that will transform a specified time series into a stationary one.
- Then, it should be possible to perform training or apply the pretrained network to the specified time series.

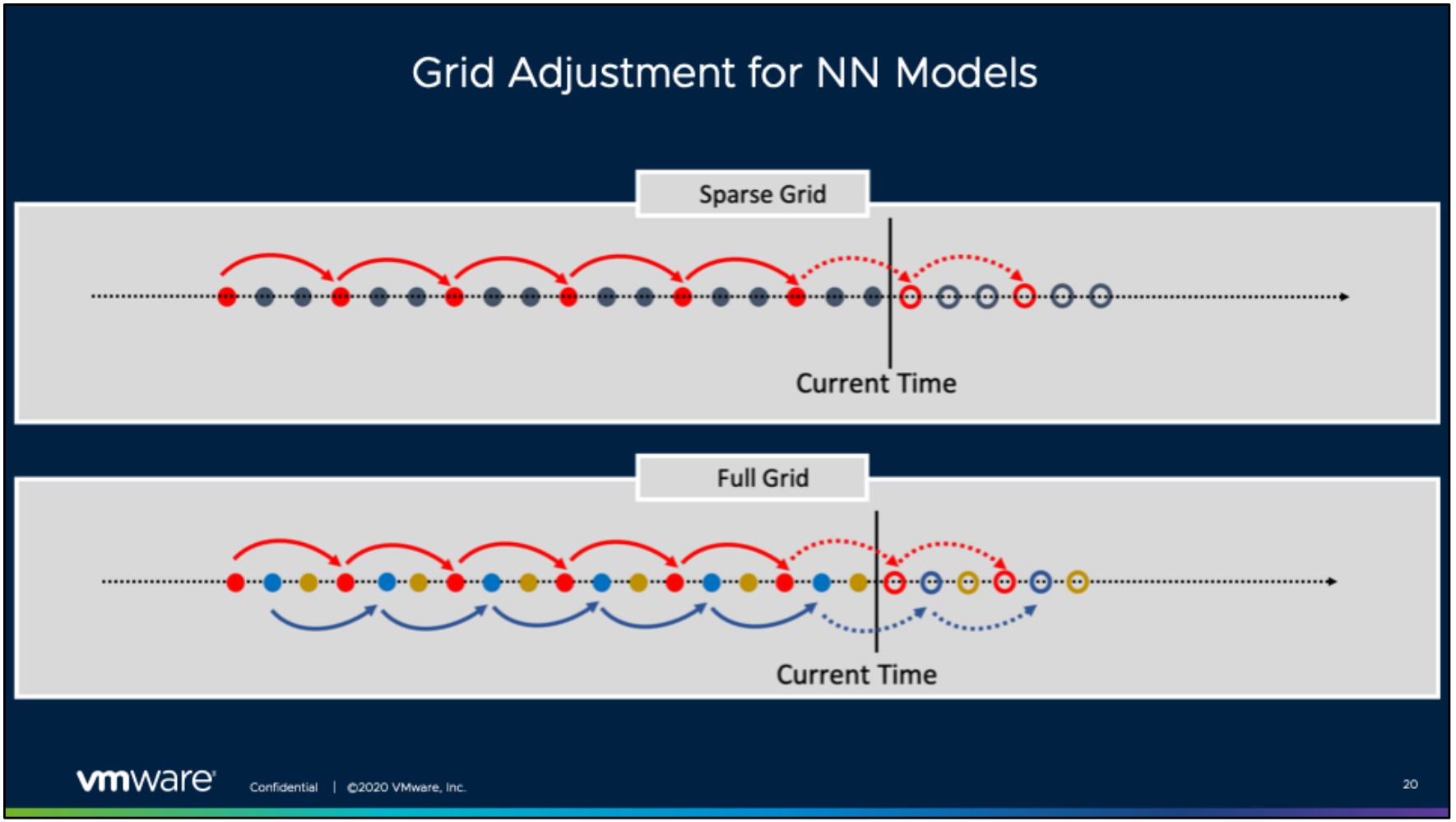
NN Model Architecture

- Training database contains 3000 time series
- NN-model uses multilayer perceptron architecture
- 2 hidden layers with 256 nodes in each
- Activation function 'relu' for the hidden layers and 'linear' for the output layer
- Input layer consists of 40 nodes and output layer of 20 points
- Overall, 81,428 weights
- 'Adam' optimizer and mean average error ('mae') as a loss function
- 5 epochs for each time series and 20 epochs for the entire database
- $batch\_size = 1500$

- We tried different models based on multilayer perceptron (MLP) and recurrent neural network (RNN) architectures.
- We decided to continue with the MLP models as found nonsignificant difference compared to RNN.
- MLP models have the simplest known architecture of NN, and they are easy to implement, train and use.



- Example of a trend-sationary data as the KPSS_ct test has the p-value bigger than 0.05.
- It is possible to transform the time series into a stationary process by detrending via linear regression. Then, after forecasting via pretrained network, the trend should be returned to get the correct forecast corresponding the “red” curve.
- The “yellow” curve corresponds to the forecast as if data is stationary – which is not correct.
- The “black” curve corresponds to the forecast as if data is stochastic trendy. It is correct, as the corresponding p-value is also bigger than 0.05.
- So data can be considered to be both from stochastic-trendy and trendy-stationary classes. However, our algorithm suggests to forecast based on trendy-stationarity having bigger priority.
- Moreover, we see that the “black” curve also performs rather accurate forecast.



- One of the problems connected with NN-models is the limited number of input and output nodes.
- As it was mentioned before, our pretrained NN-model forecasts 20 points via 40 historical points.
- This is very strong limitation that maybe will allow to handle a trend rather well and missing smaller peculiarities especially for longer historical periods.
- We decided to take historical grids with number of points multiple to 40, divide the entire grid into sub-grids with 40 points in each and sequentially fed the NN model.
- For each of those sub-grids, the NN-model will provide with 20 forecasts which we will regroup together to get the forecast with denser resolution.

Seasonality Test with Phase Dispersion Minimization (PDM)

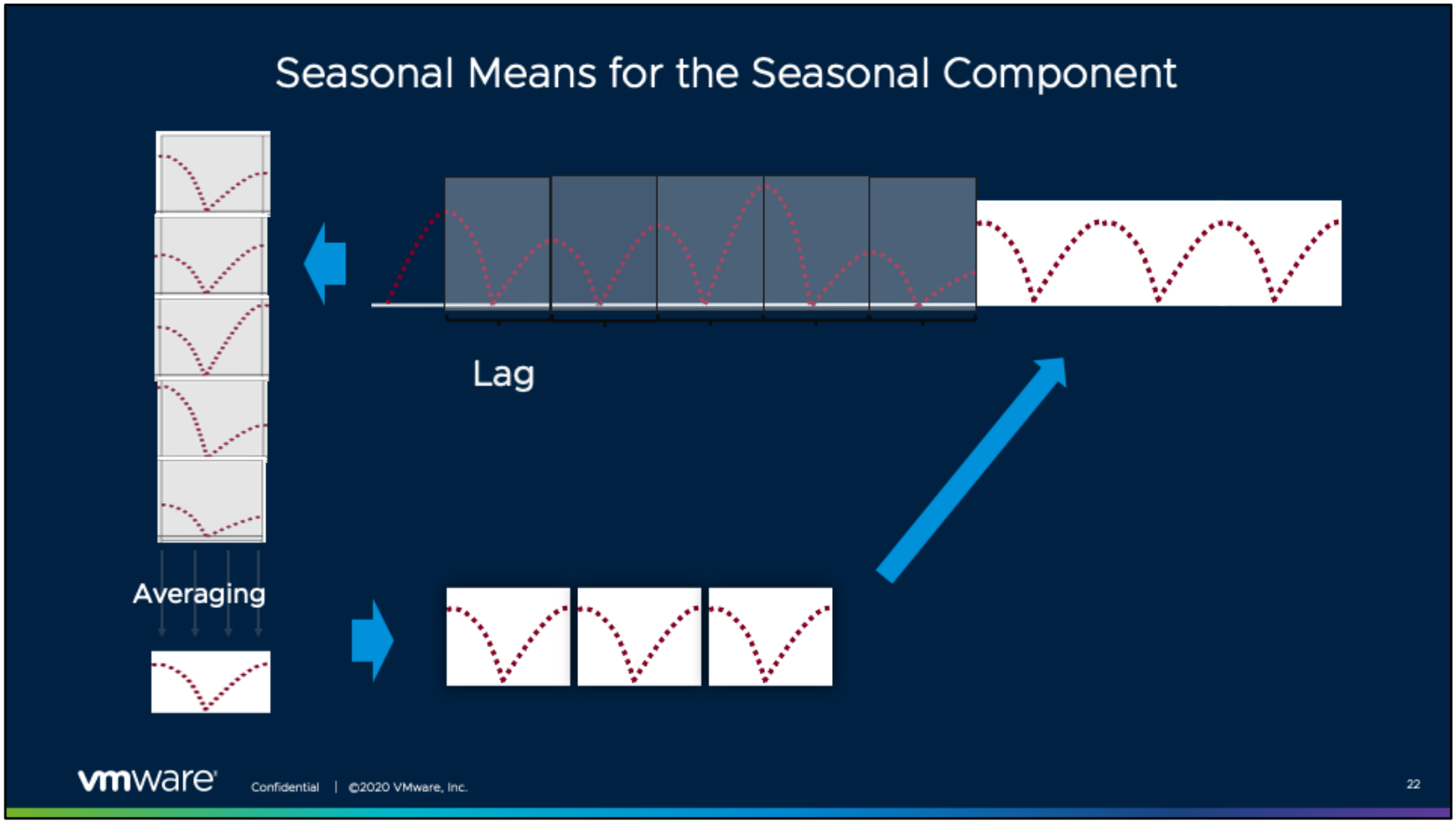
$$Time\ Series = Seasonal\ Component + Residual$$

$$Importance(lag) = 1 - \frac{Var(Residual)}{Var(Seasonal\ Component + Residual)}$$

Deterministic seasonality for $lag = lag_0$, if

$$Importance(lag_0) > 0.6$$

- For seasonal-trend decomposition see https://www.wessa.net/download/stl.pdf. It describes powerful STL decomposition invented in: Cleveland, R. B., Cleveland, W. S., McRae, J. E., & Terpenning, I. J. (1990). STL: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics*, *6*(1), 3–33.
- We use much simpler decompositions like linear trend + seasonal means + residuals, or without detrending like seasonal means + residuals. Comparison will show is it worth to apply detrending or not.



- Seasonal means is the simplest and the most obvious approach for time series seasonal component.
- It requires cutting data into portions with the length = lag (say weekly, starting from the end), collecting those portions together and calculating the average time series with length = lag (seasonal means).
- Then, we replicate it as necessary (k*lag) to get time series which length exactly equals to the length of the initial time series – this is what is known as seasonal component.
- In average, it mimics seasonal patterns in data if available.
- Importance measure will show the strength of those patterns compared to the residual time series for a selected lag.
- We repeat the procedure for different lags and decide wether data is periodic or not and if yes, what is the period.

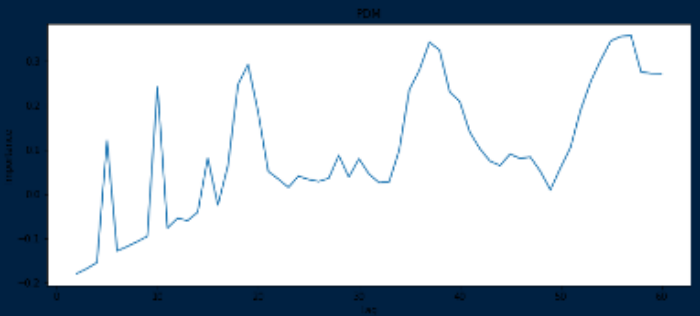
Why PDM?

- Applicable for time series with a few observations
- Applicable for data with non-regular sampling
- Applicable for data with gaps
- Captures non-sinusoidal periodicities
- Fast implementation for periodic data as the procedure can be stopped if the corresponding lag is detected

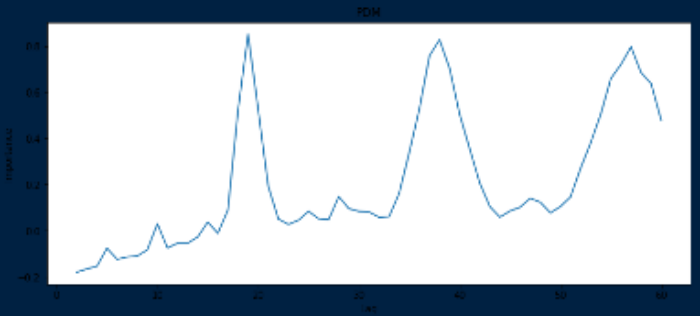
Details see also in:
Stellingwerf, R.F.: Period Determination using Phase Dispersion Minimization. The Astrophysical Journal 224, 953-960 (1978).

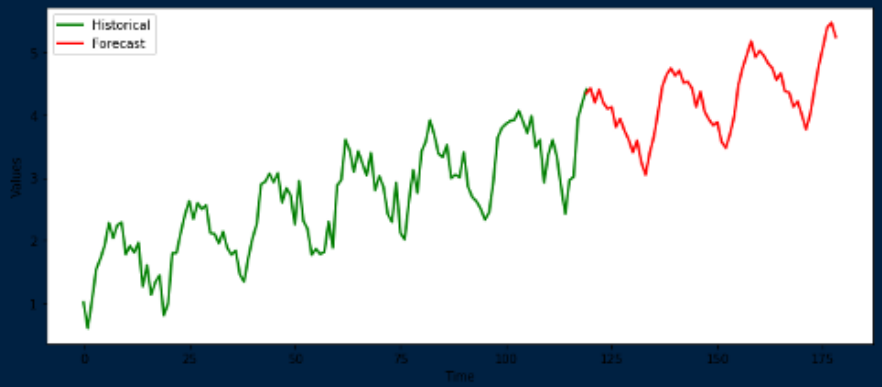
Example of a Trendy-Periodic Data

Lag = 57, Importance = 0.358 – without detrending

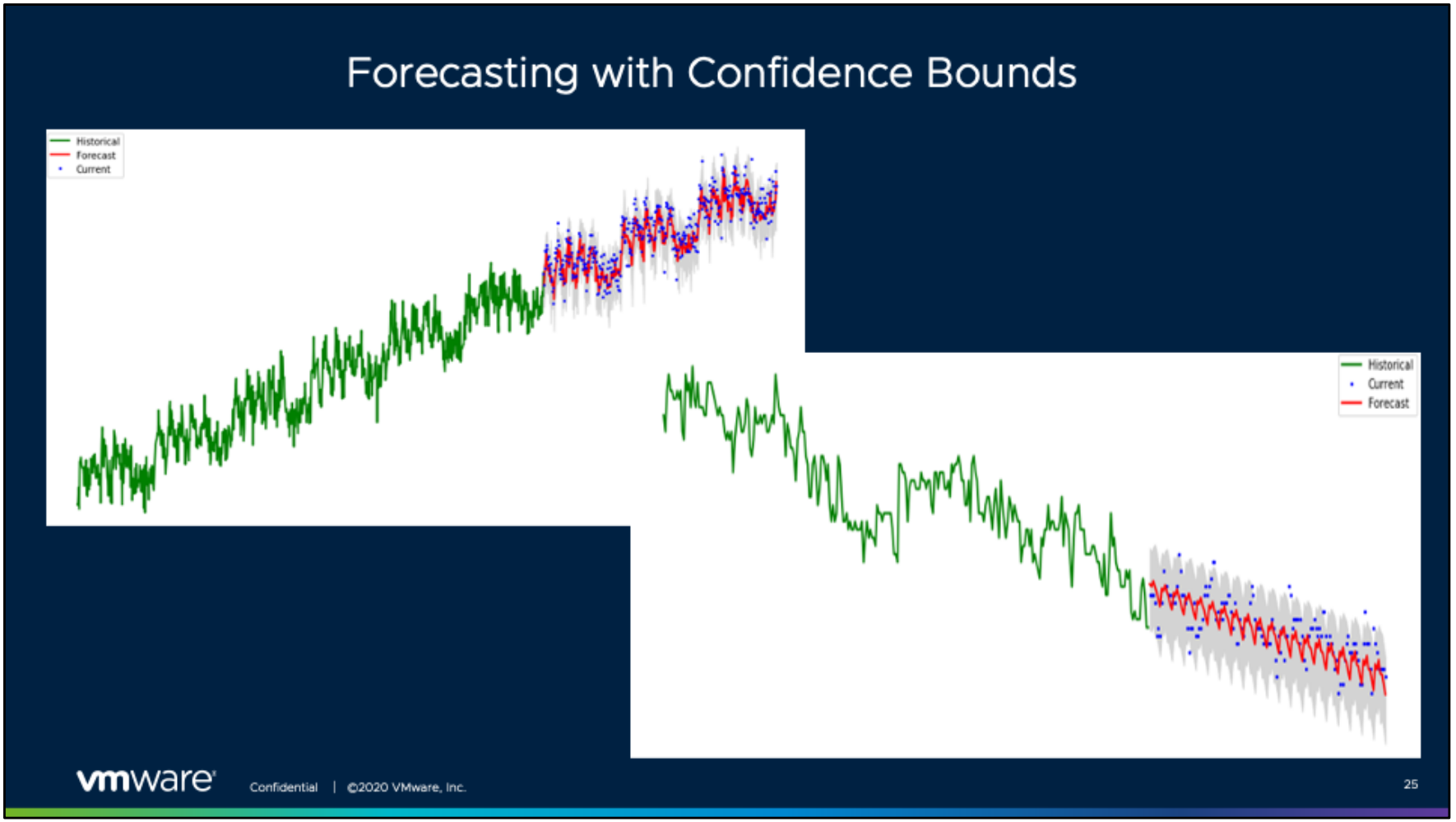


Lag = 19, Importance = 0.851 – after detrending





- The right figure shows an example of a trendy-periodic data (the “green” curve).
- Firstly, we calculate importance of different lags without detrending. Top-left figure shows that the maximum importance corresponding to lag=57 is only 0.36. Data is not periodic without detrending.
- Secondly, we detrend data by the linear regression and calculate importance of lags for the detrended data. Bottom-left figure shows that lag = 19 corresponds to the maximum importance with importance = 0.86. Data is trendy-periodic.
- We remove trend, then remove seasonality by the seasonal means with lag = 19, and apply NN-model on the residual time series. Afterwards, we restore seasonality and trend. The “red” curve of the right figure shows the corresponding forecast.



Confidence bounds are necessary for any forecasting approach.

Wavefront AI Genie UI – a Trendy Time Series

A user can select:

- Time series data
- Forecast horizon
- The sensitivity of confidence bounds

Details see in:
https://docs.wavefront.com/ai_genie.html

Wavefront AI Genie UI – a Stationary Time Series

Details see in:
https://docs.wavefront.com/ai_genie.html