CentraleSupélec

université PARIS-SACLAY

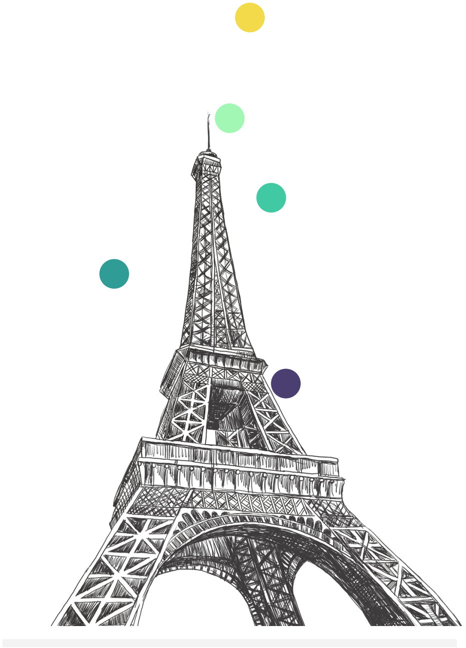
Using Machine Learning models to predict air pollution around Eiffel Tower

Adina Bondoc, Jezuela Gega

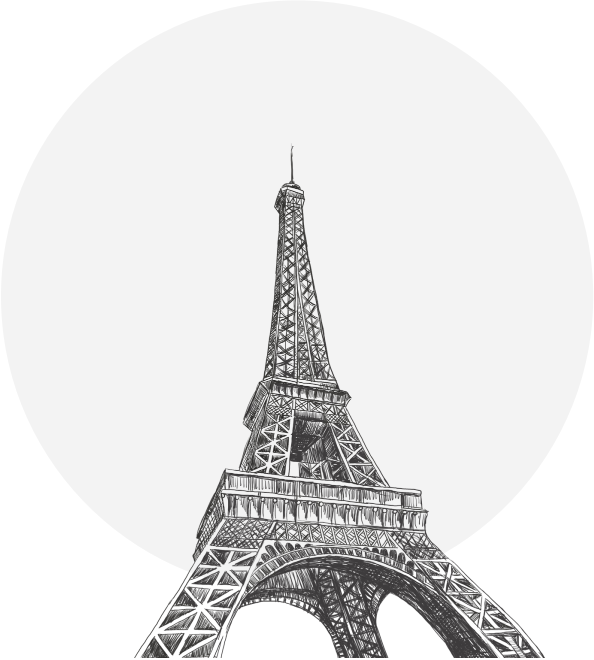
Supervised by Tom Dupuis

January, 2024

Table of contents



Objective



The aim of this project is to model the **pollutant concentrations** in time using ML models around Eiffel tower.

Problem

Air pollution is the most considerable environmental health risk in all of Europe (European Environment Agency (EEA), 2022).

To tackle pollution, European Union (EU) came up with the approach in 2008 to measure the air quality in areas where people are affected adversely. [1]

Long-term exposure to high levels of **nitrogen dioxide can cause chronic lung disease**. It may also affect the senses, for example, by reducing a person's ability to smell an odour. [2]

[1] A. Samad, A. (2023, July 28). Air pollution prediction using machine learning techniques – an approach to replace existing monitoring stations with Virtual Monitoring Stations. Atmospheric Environment. https://www.sciencedirect.com/science/article/pii/S1352231023004132#bib14
[2] Queensland;, c=AU; o=The S. of. (2017, April 10). Air Pollution. Queensland Government. https://www.qld.gov.au/environment/management/monitoring/air/air-pollution

Problem

Air quality guidelines

The Environmental Protection (Air) Policy 2019 (EPP Air) objectives for nitrogen dioxide are:

- 0.12 parts per million (ppm) for a 1-hour exposure period
- 0.03ppm for an annual exposure period.

The National Environment Protection (Ambient Air Quality) Measure standards for nitrogen dioxide are:

- 0.08ppm for a 1-hour exposure period
- 0.015ppm for an annual exposure period.

[3]

[3] ScienceDaily. (2020, June 26). *Traffic Density, wind and air stratification influence concentrations of Air Pollutant No2*. ScienceDaily. https://www.sciencedaily.com/releases/2020/06/200626114750.htm

Our Dataset

DateTime	NO2	NO	NOX
2022-01-01 00:00:00+00:00	13.3	2.3	16.8
2022-01-01 01:00:00+00:00	10.1	0.9	11.5
2022-01-01 02:00:00+00:00	7.5	0.3	8
2022-01-01 03:00:00+00:00	6.1	0.2	6.4
2022-01-01 04:00:00+00:00	5.9	0.2	6.2

Source: https://data-airparif-asso.opendata.arcgis.com/datasets/2023-pa07

Our Dataset

- Location: Eiffel tower
- From January 2018 until January 2024
- Time series - hourly data
- NOX ≈ NO2+NO

Source: https://data-airparif-asso.opendata.arcgis.com/datasets/2023-pa07

Our Dataset

It was shown that **wind** speed and the height of the lowest air layer are the most important factors that determine how much pollutants can accumulate locally.

(Traffic density, wind and air stratification influence concentrations of air pollutant NO2: https://www.sciencedaily.com/releases/2020/06/200626114750.htm)

Source: https://data-airparif-asso.opendata.arcgis.com/datasets/2023-pa07

Our Dataset

DateTime	NO2	Temp	Wind
2022-01-01 00:00:00+00:00	13.3	22	16.8
2022-01-01 01:00:00+00:00	10.1	21	11.5
2022-01-01 02:00:00+00:00	7.5	20	8
2022-01-01 03:00:00+00:00	6.1	22	6.4
2022-01-01 04:00:00+00:00	5.9	18	6.2

Source: https://data-airparif-asso.opendata.arcgis.com/datasets/2023-pa07

PreProcessing

Handling Missing Values:

Linear imputation + backfill

Data Normalization:

Normalizes values in 0-1 scale

Seasonal Adjustment

Creation of binary variables related to dates (month, holidays, day, weekdays, etc.)

Train-test Split

Training: July 2021 - June 2023
Validation: July 2023 - December 2023
Test: January 2024

Our Models

1 ARIMA

Statistical method used for time series models

- *AutoRegressive*: past values
- *Integrated*: differencing for stationarity
- *Moving Average*: trends & patterns

2 Prophet

Developed by Facebook for time series based on an additive model

- Fast & Easy
- Best for strong seasonality
- Robust for outliers and missing data
- Holiday effects

3 RNN

Recurrent Neural Networks

- Sequential data
- Maintains a hidden state/internal memory

Vanilla

- Short term dependencies
- Vanishing gradient problem
- Difficult for long term

LSTM

Long short-term memory networks

- Extends memory of RNNs
- Can read, write and delete information

Architectures

ARIMA

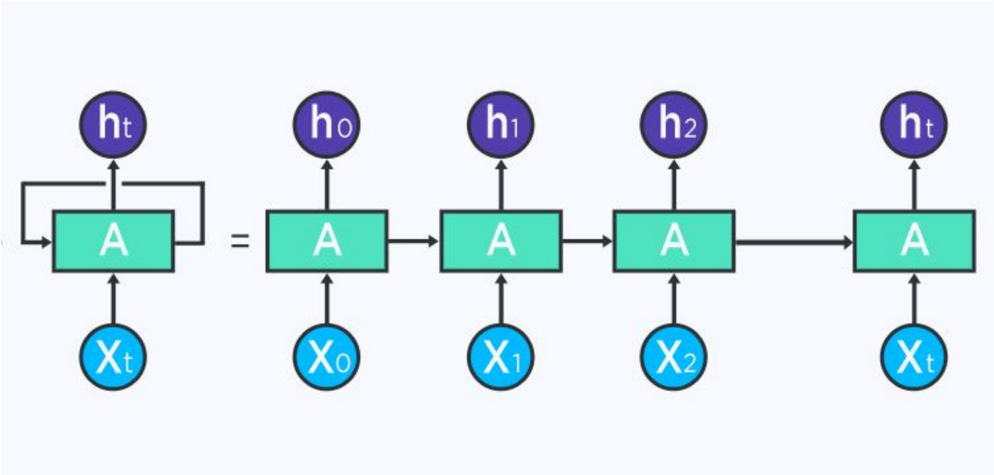
$$\underbrace{y_t^{(d)}}_{\text{Integrated}} = c + \varepsilon_t + \underbrace{\phi_1 y_{t-1}^{(d)} + \phi_2 y_{t-2}^{(d)} + \ldots + \phi_p y_{t-p}^{(d)}}_{\text{Auto-Regressive}} + \underbrace{\theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \ldots + \theta_p \varepsilon_{t-q}}_{\text{Moving Average}}$$

Prophet

$$y(\mathrm{t}) = \mathit{trend}(\mathrm{t}) + \mathit{seasonality}(\mathrm{t}) + \mathit{holidays}\,(\mathrm{t}) + \mathit{error}(\mathrm{t})$$

https://mlpills.dev/time-series/introduction-to-arima-models/
https://peerj.com/preprints/3190/

Architectures: Vanilla RNN



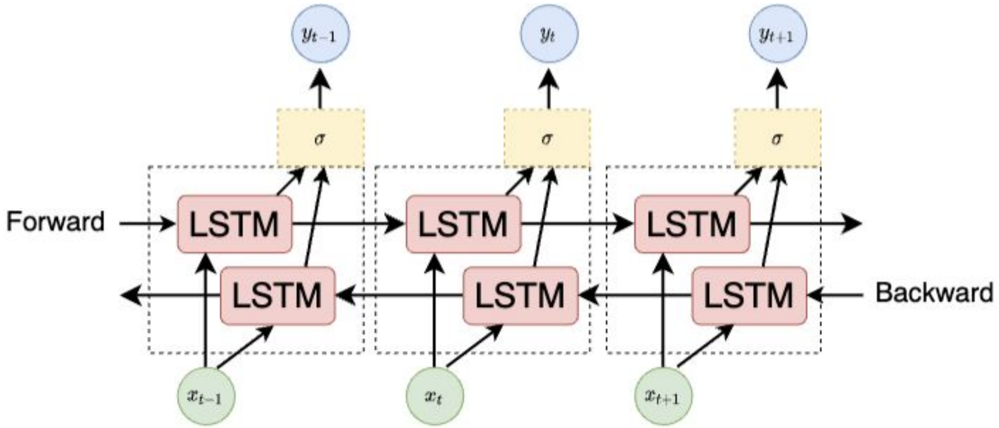
Dropout after every layer

Dropout after first layer

Batch Normalization after first layer

https://builtin.com/data-science/recurrent-neural-networks-and-lstm

Architectures: LSTM



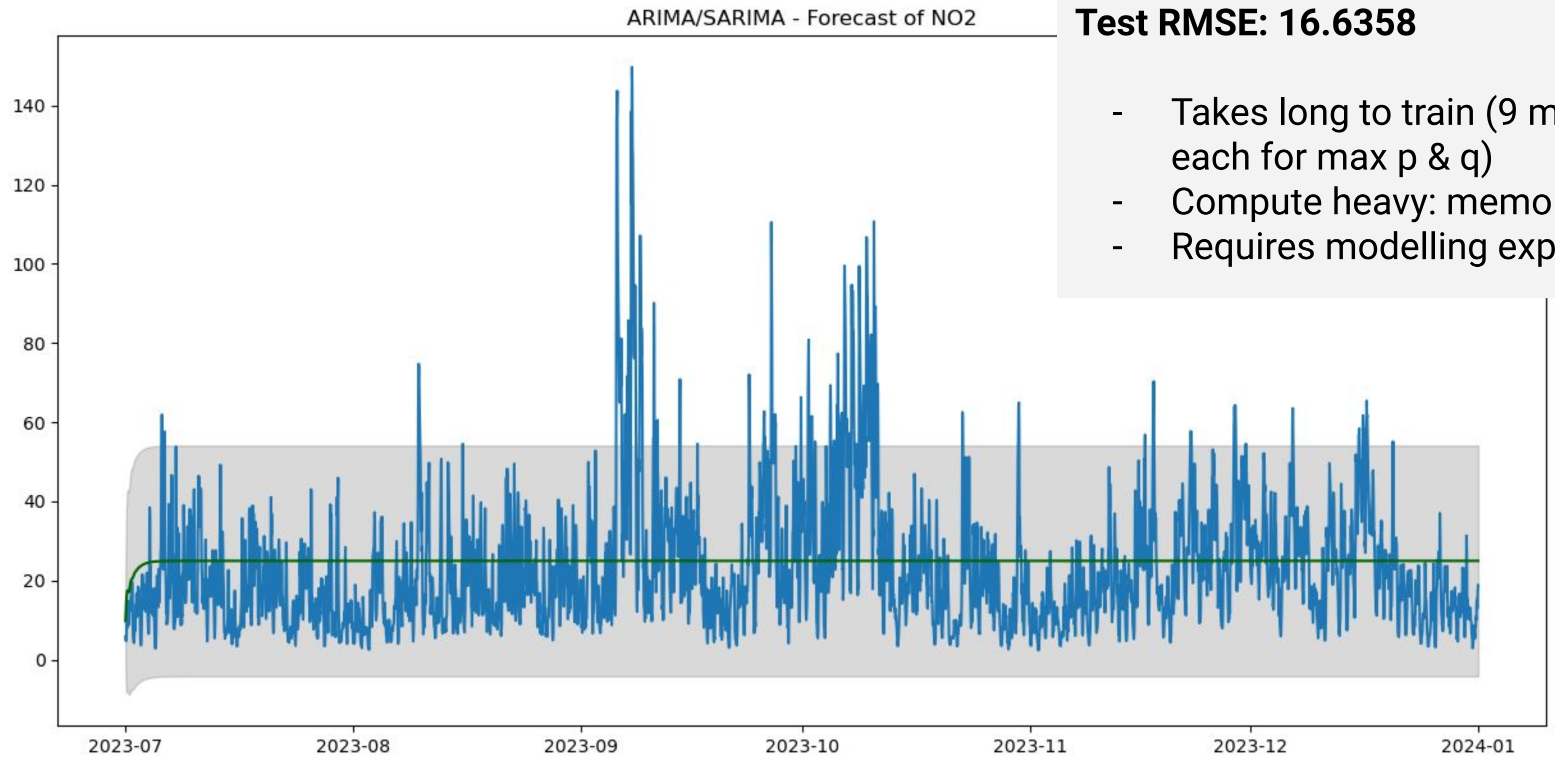
Univariate & Multivariate

50+ hidden layers

Only forward & Bidirectional

Source:https://www.researchgate.net/publication/351917573_Predicting_COVID-19_cases_using_bidirectional_LSTM_on_multivariate_time_series, page 6

Results: ARIMA (11, 0, 0)

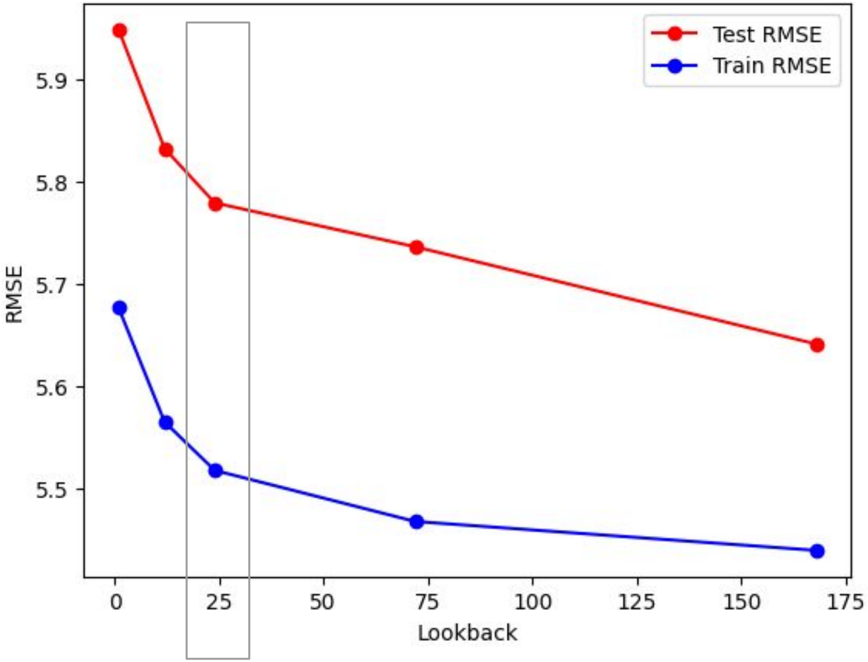


Test RMSE: 16.6358

- Takes long to train (9 mins, 24 each for max p & q)
- Compute heavy: memory error
- Requires modelling expertise

Results: Prophet

Univariate



Lookback

[1, 12, 24, 72, 168]

Features

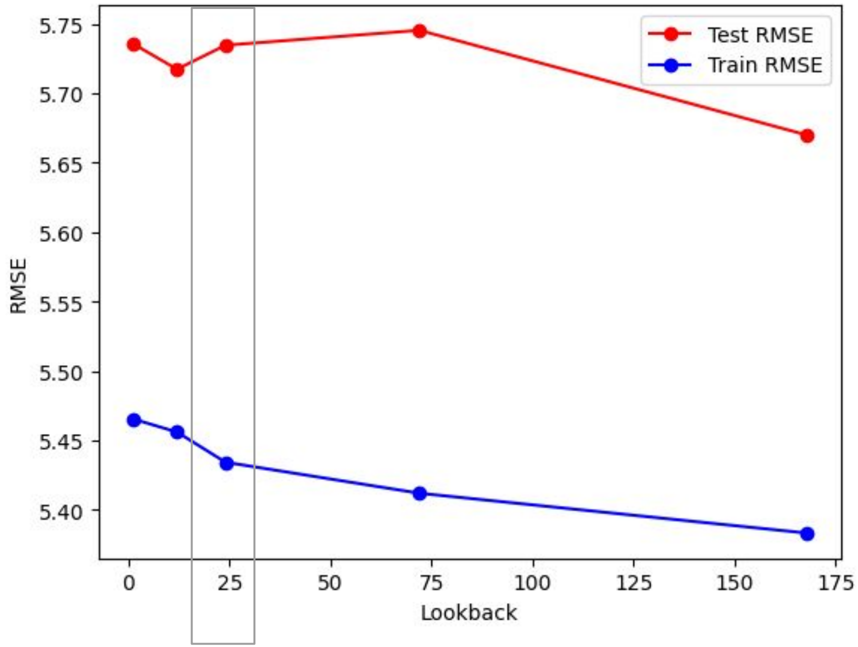
Univariate, Multivariate: Wind & Temp

Best Model:

Lookback = 24
Train RMSE = 5.5176
Val RMSE = 5.7788

Results: Prophet

Multivariate: Wind & Temp



Lookback

[1, 12, 24, 72, 168]

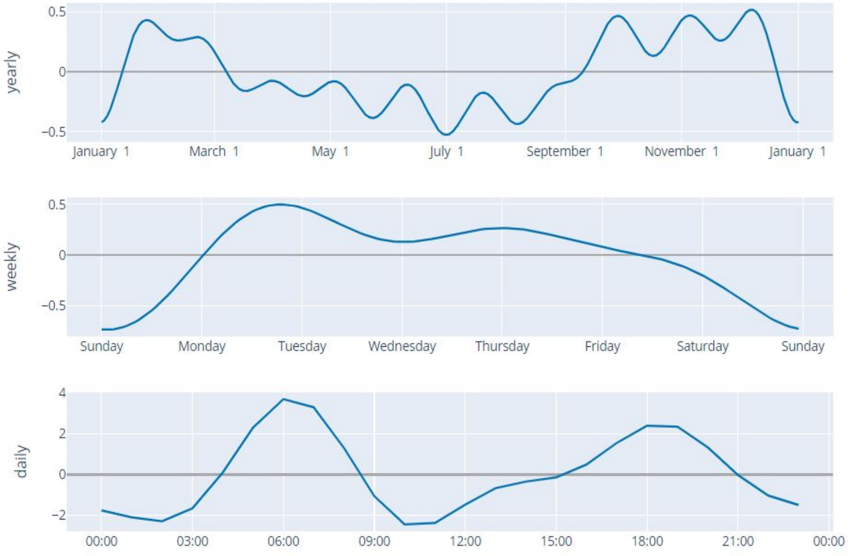
Features

Univariate, Multivariate: Wind & Temp

Best Model:

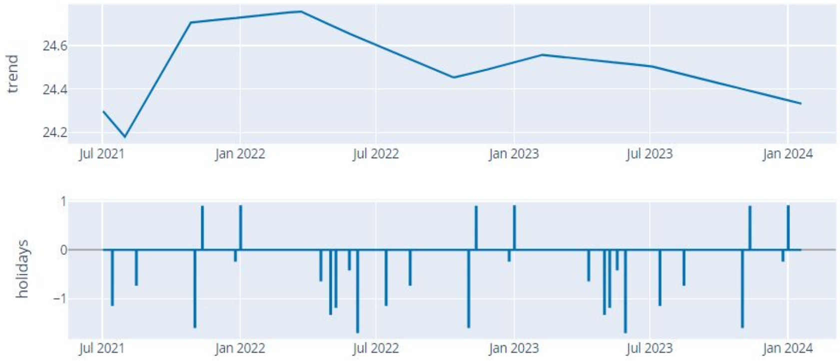
Lookback = 24
Train RMSE = 5.3834
Val RMSE = 5.6699

Results: Prophet Best



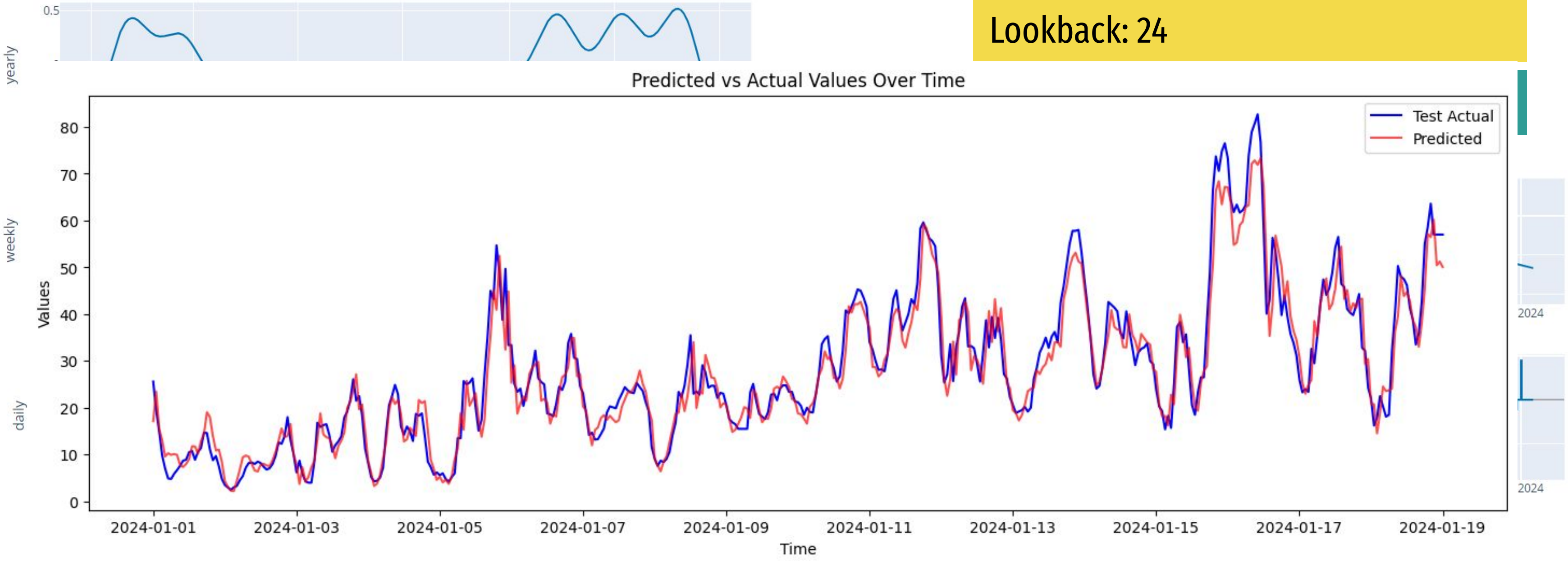
Lookback: 24

Features: Univariate



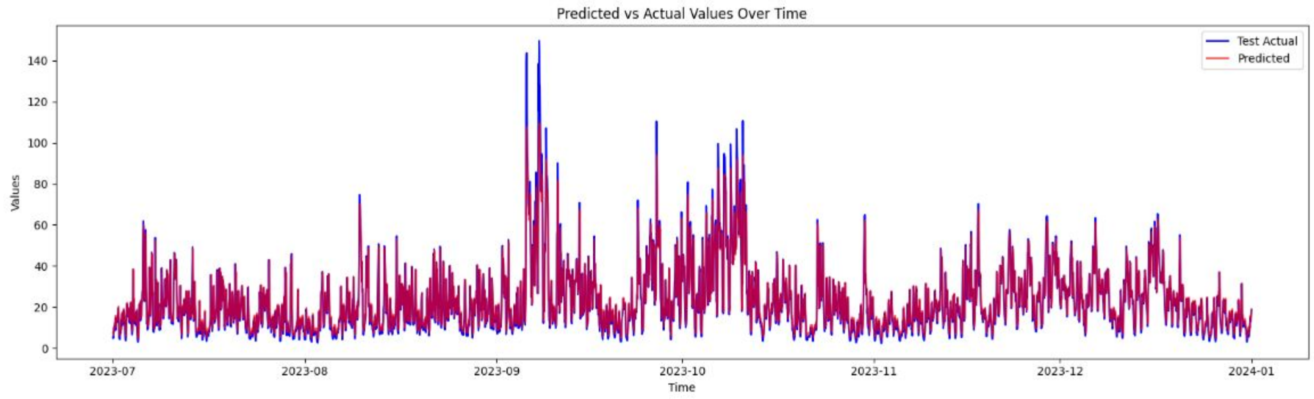
Results: Prophet Best

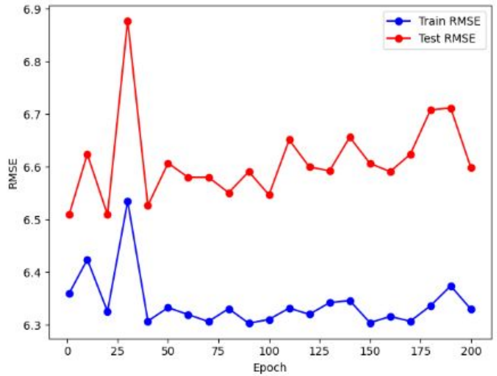
Lookback: 24

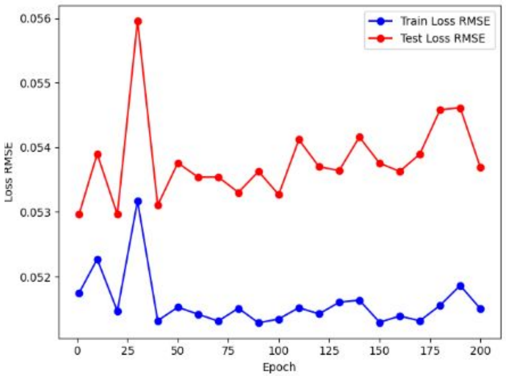


Train RMSE = 5.5604
Test RMSE = 4.3148

Results: RNN Univariate







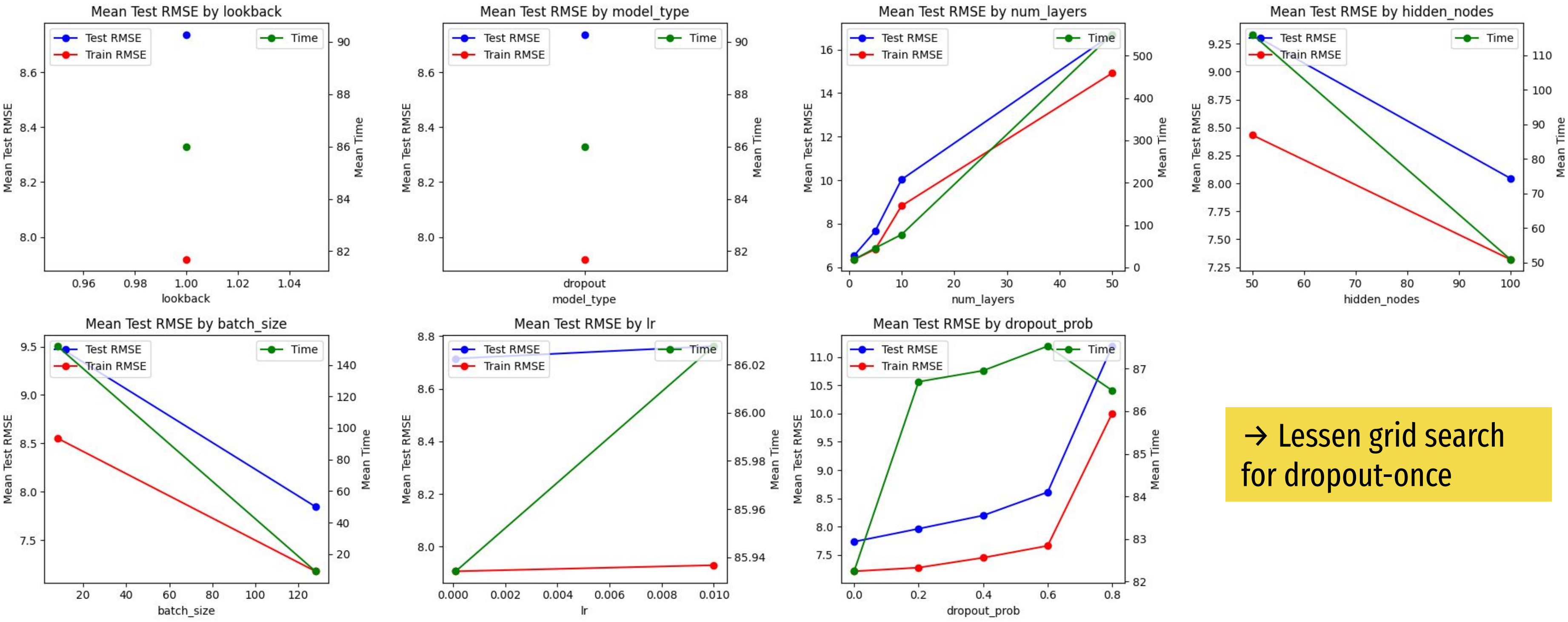

```
1 layer, no dropout, 1 hour lookback
Epoch 200
train RMSE 6.3298, val RMSE 6.5984
train loss RMSE 0.0515, val loss RMSE 0.0537
```

→ Use only 10 epochs for training

Results: RNN Univariate Variations

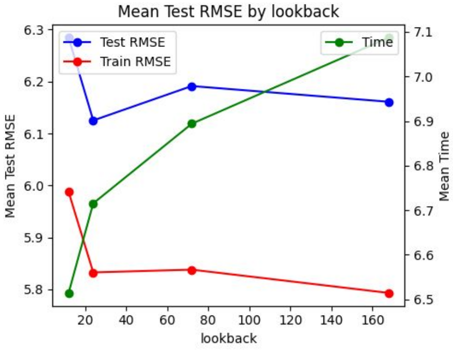
```
iter_lookback = 1
iter_model_type = 'dropout'
iter_num_layers = [1, 5, 10, 50]
iter_hidden_nodes  = [50, 100]
iter_batch_size = [8, 128]
iter_lr = [0.01, 0.0001]
iter_dropout_prob = [0, 0.2, 0.4, 0.6, 0.8]
```

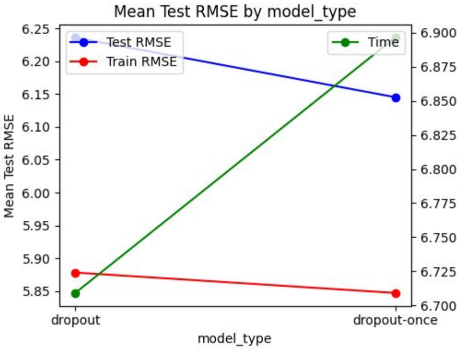
Results: RNN Univariate Variations



→ Lessen grid search for dropout-once

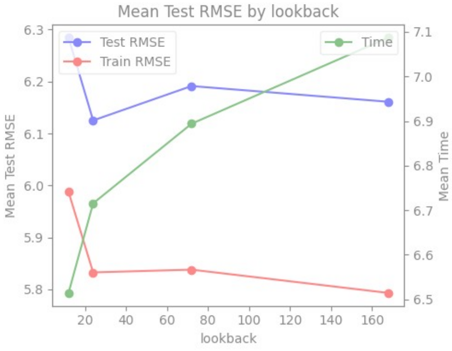
Results: RNN Univariate Variations

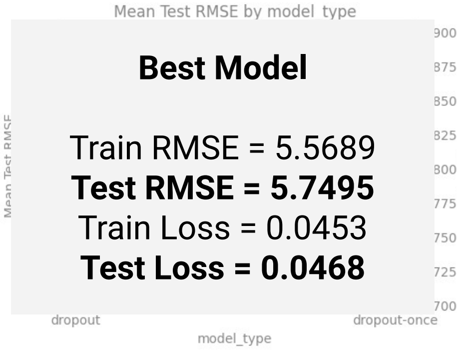





```
iter_lookback = [12, 24, 72, 168]
iter_model_type = ['dropout', 'dropout-once']
iter_num_layers = [1, 5]
iter_hidden_nodes  = [100]
iter_batch_size = [128]
iter_lr = [0.0001]
iter_dropout_prob = [0, 0.2, 0.4, 0.6]
```

Results: RNN Univariate Variations





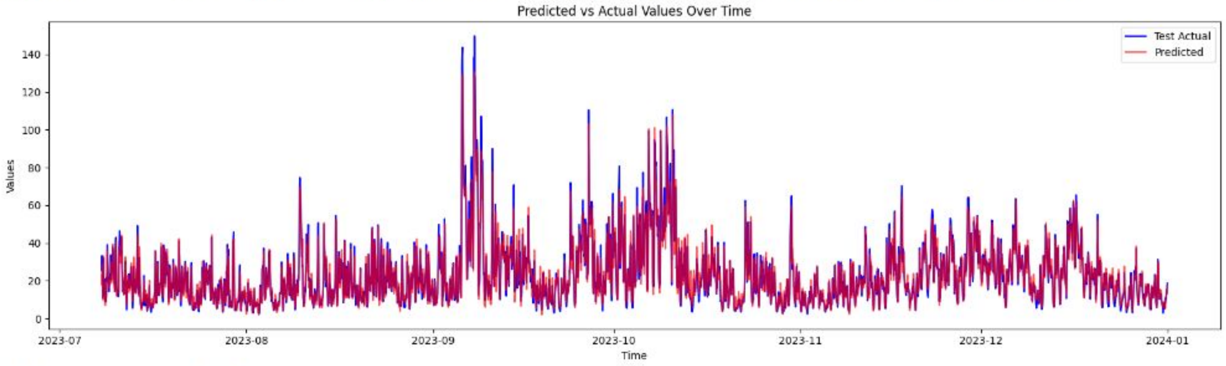
Best Model

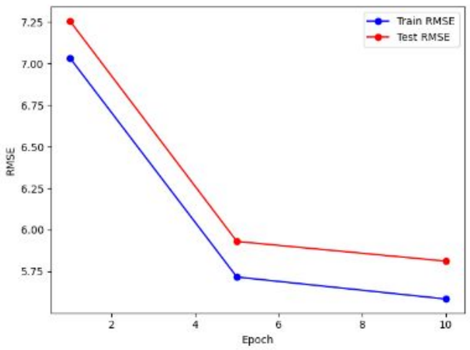
Train RMSE = 5.5689
Test RMSE = 5.7495
Train Loss = 0.0453
Test Loss = 0.0468

```
'lookback': 168,
'model_type': 'dropout-once',
'num_layers': 1,
'hidden_nodes': 100,
'batch_size': 128,
'lr': 0.0001,
'dropout_prob': 0.0
```

→ Try best univariate model parameters with exogenous variables

```
Epoch 1         || train RMSE 7.0339, test RMSE 7.2566 || train loss RMSE 0.0572, test loss RMSE 0.0590
Epoch 5         || train RMSE 5.7166, test RMSE 5.9305 || train loss RMSE 0.0465, test loss RMSE 0.0483
Epoch 10        || train RMSE 5.5844, test RMSE 5.8122 || train loss RMSE 0.0454, test loss RMSE 0.0473
```



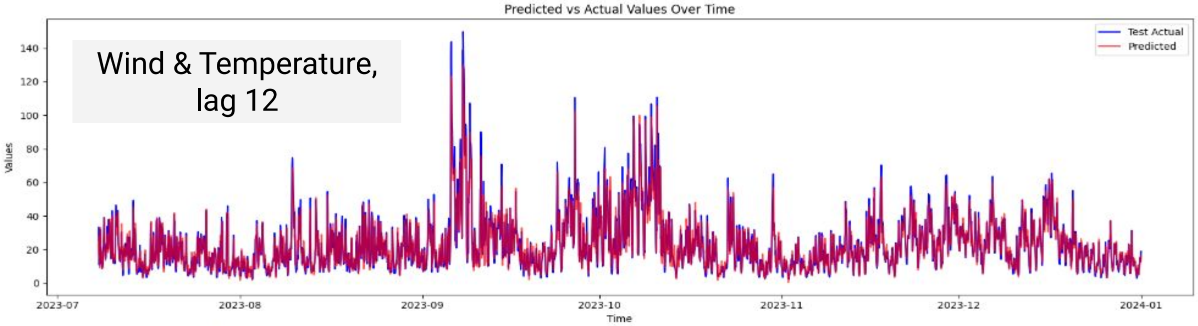


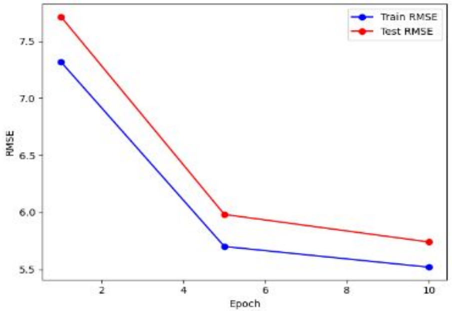

```
RMSE on Test Set: 5.8122
```

Results: RNN Multivariate

```
Epoch 1     || train RMSE 7.3179, test RMSE 7.7142 || train loss RMSE 0.0595, test loss RMSE 0.0628
Epoch 5     || train RMSE 5.7007, test RMSE 5.9815 || train loss RMSE 0.0464, test loss RMSE 0.0487
Epoch 10    || train RMSE 5.5196, test RMSE 5.7404 || train loss RMSE 0.0449, test loss RMSE 0.0467
```

Wind & Temperature, lag 12



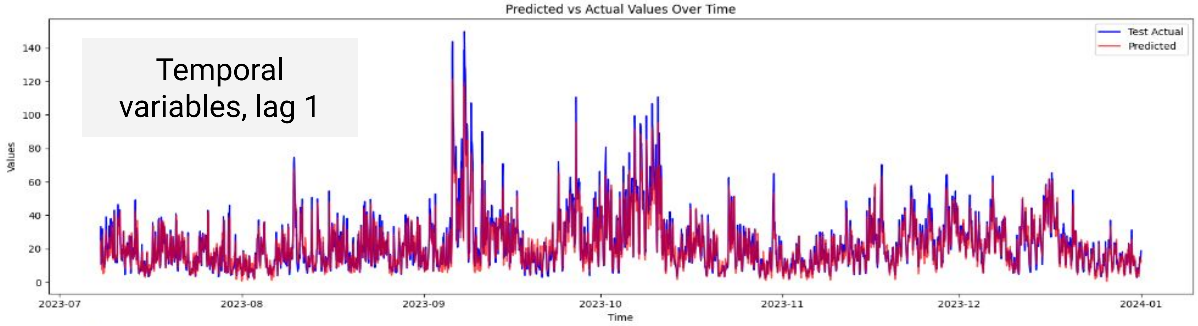


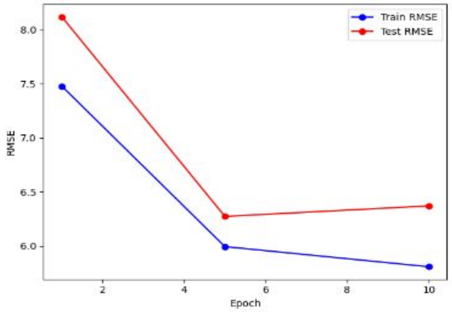

```
RMSE on Test Set: 5.7404
```

→ Is it worth it?

```
Epoch 1     || train RMSE 7.4795, test RMSE 8.1214 || train loss RMSE 0.0609, test loss RMSE 0.0661
Epoch 5     || train RMSE 5.9938, test RMSE 6.2736 || train loss RMSE 0.0488, test loss RMSE 0.0510
Epoch 10    || train RMSE 5.8075, test RMSE 6.3708 || train loss RMSE 0.0473, test loss RMSE 0.0518
```

Temporal variables, lag 1



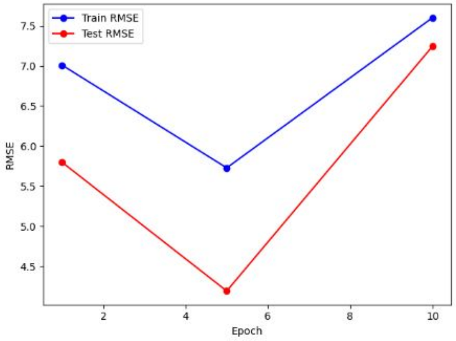


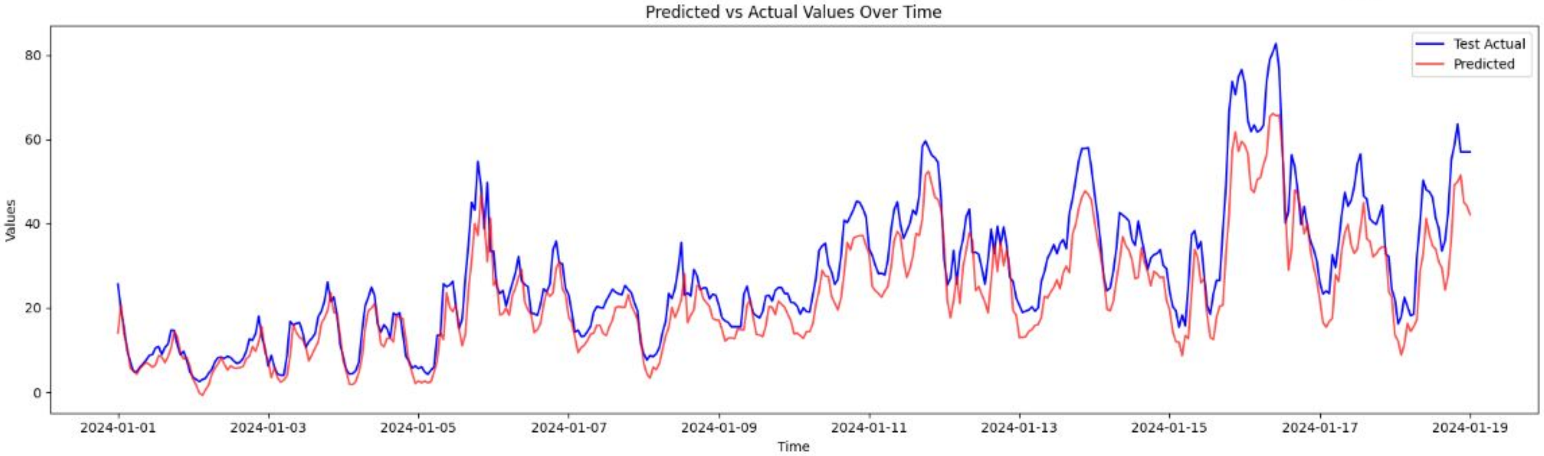

```
RMSE on Test Set: 6.3708
```

Results: RNN Best Model Rerun

```
'lookback': 168,
'model_type': 'dropout-once',
'num_layers': 1,
'hidden_nodes': 100,
'batch_size': 128,
'lr': 0.0001,
'dropout_prob': 0.0
```

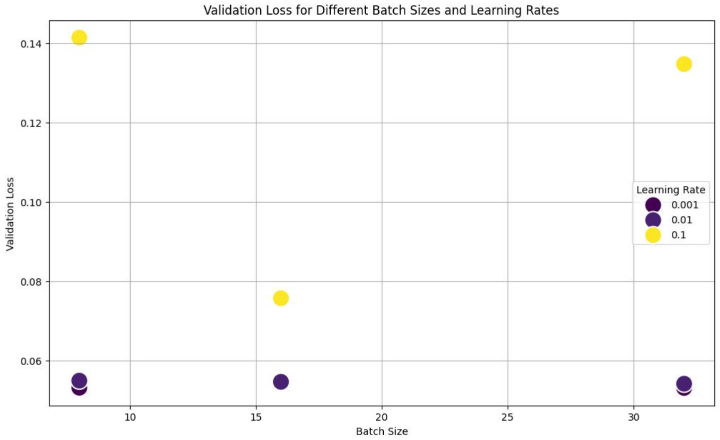
Train RMSE = 7.6005
Test RMSE = 7.2446
Train Loss = 0.0513
Test Loss = 0.0489

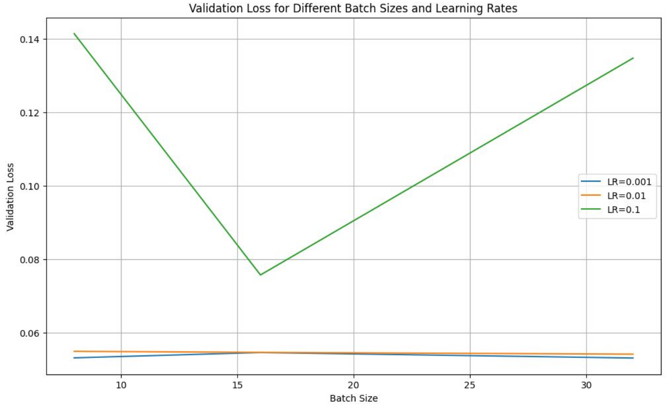




Results: LSTM - Univariate

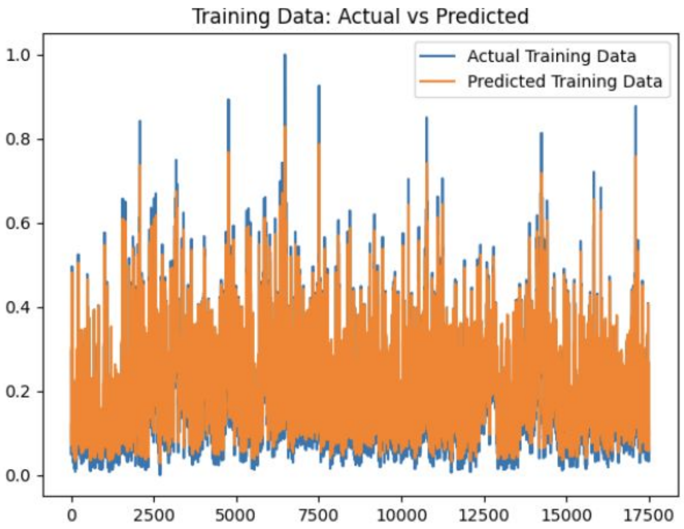
Tuning LSTM:
batch_sizes = [8, 16, 32]
learning_rates = [0.001, 0.01, 0.1]

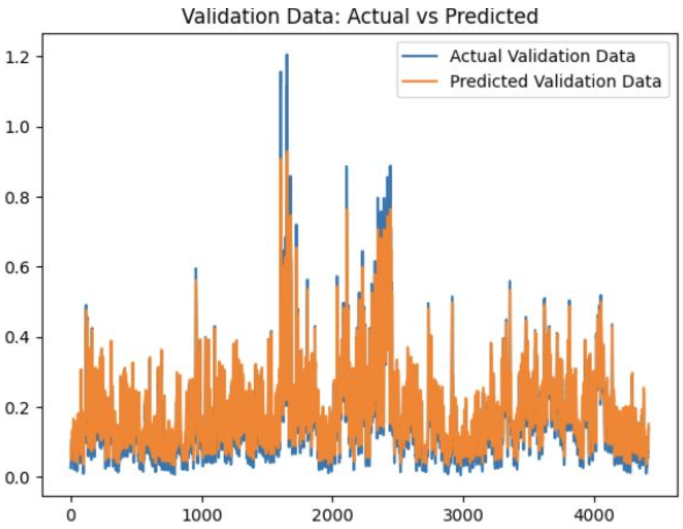




Results: LSTM - Univariate

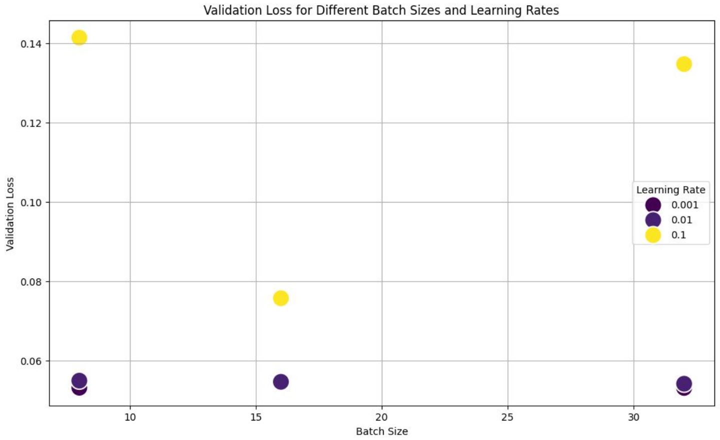
Best parameters:
Batch Size 32
Learning Rate 0.001000
Validation Loss 0.053087

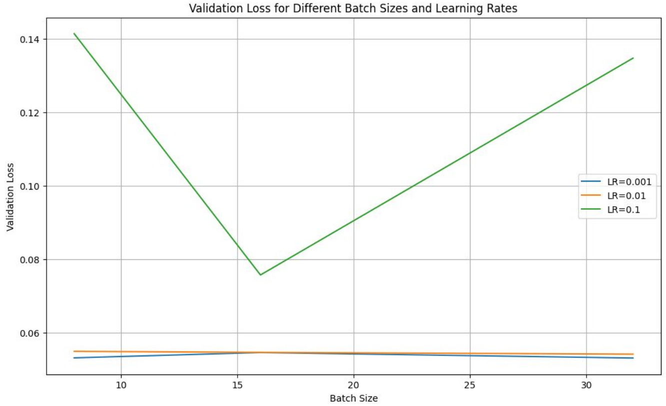




Results: LSTM - Multivariate

Tuning LSTM:
num_layers_options = [1, 2, 3]
hidden_units_options = [8,32, 64]
learning_rate_options = [0.001, 0.01, 0.1]
batch_size_options = [8, 16, 32]

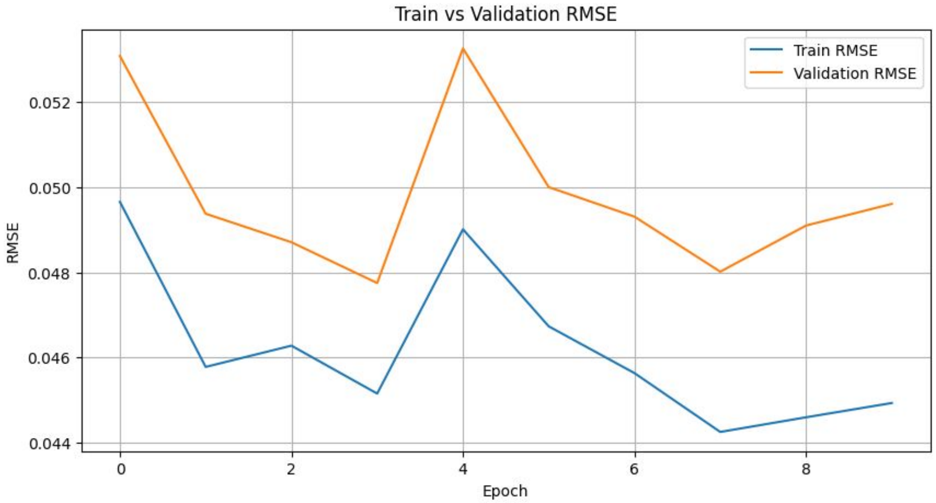




Results: LSTM - Multivariate

With dropout - 10 epochs

```
Input_size=55
hidden_size=50
num_layers=1
```



Final: Train RMSE 0.0449, Validation RMSE 0.0496

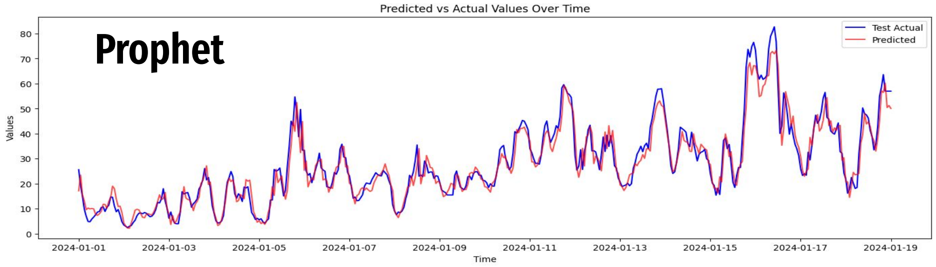
Results: LSTM - Multivariate

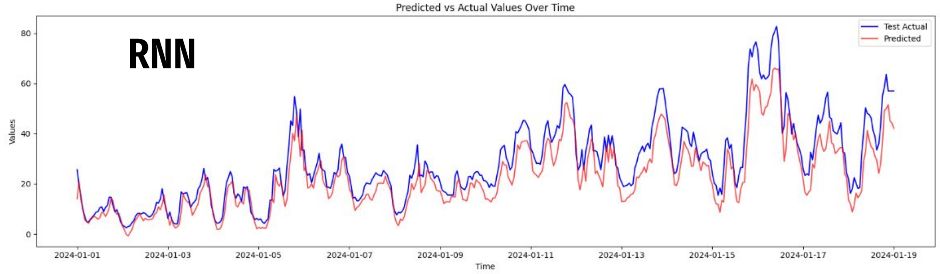
No dropout - 10 epochs

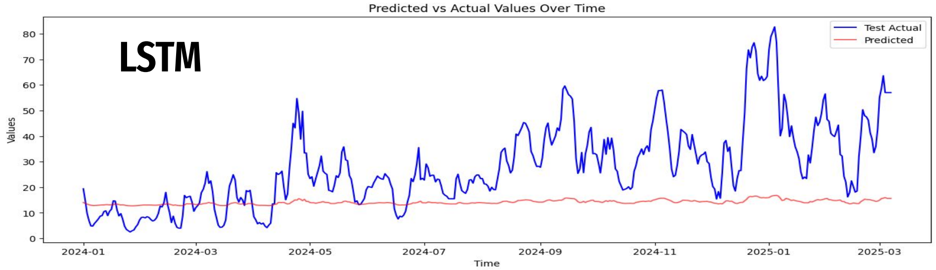
```
input_size=55
hidden_size=50
num_layers=1
```

`Final: Train RMSE 0.0437, Validation RMSE 0.0472`

Results: Comparison of Best Models



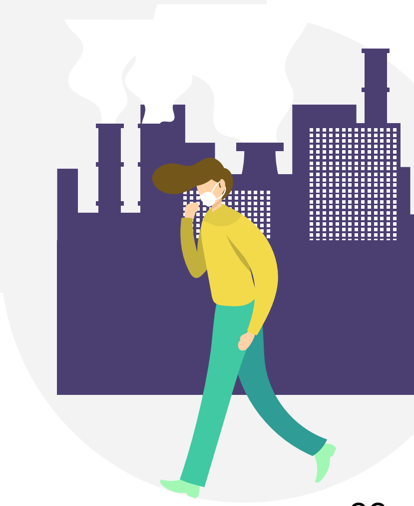




	Train RMSE	Test RMSE
Prophet	5.5604	4.3148
RNN	7.6005	7.2446
LSTM	17.9568	20.7277

Limitations

- GPU possible for around 1 hour for free in Colab
- Limited number of combinations for hyperparameter tuning
- Limited number of epochs - 10 epochs were used
- Data - 2020 disruption of the trend



Conclusions

- Deep learning requires a lot of hyperparameter tuning to get good results
- Lookback most effective in improving the model performance
- Look into: feature selection for better multivariable models
- Prophet - optimized for time series modelling
- LTSM overfits with small amount of data even if the hyperparameters a properly tuned