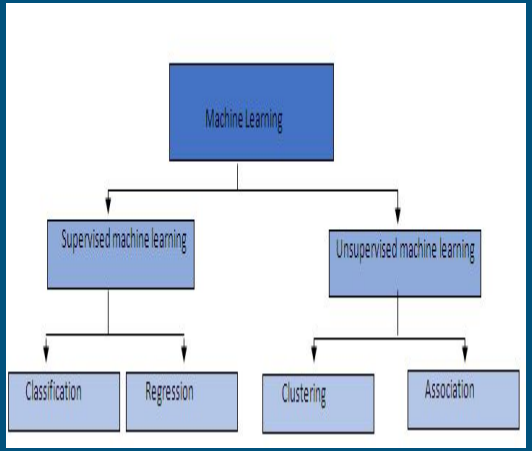
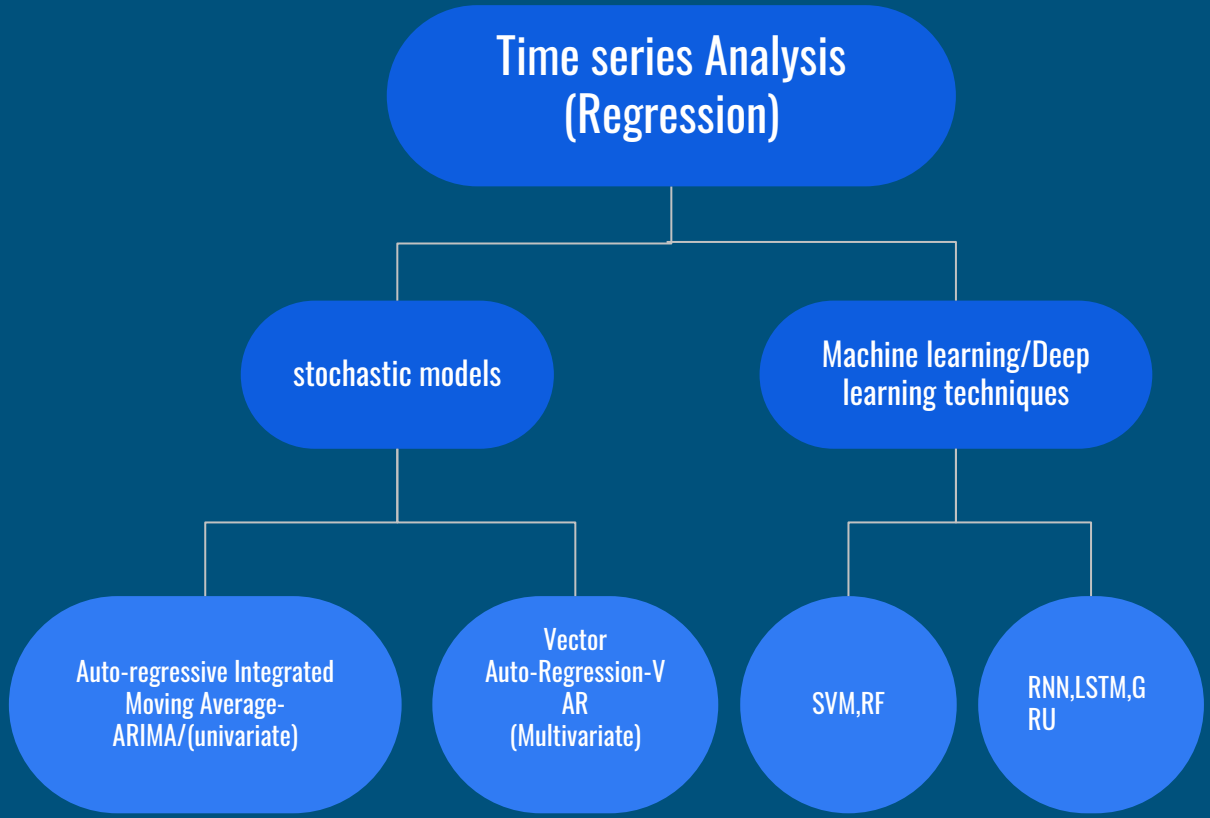
Project 1: Multivariate Time Series Forecasting Using Deep Learning

Presenter: Most Husne Jahan

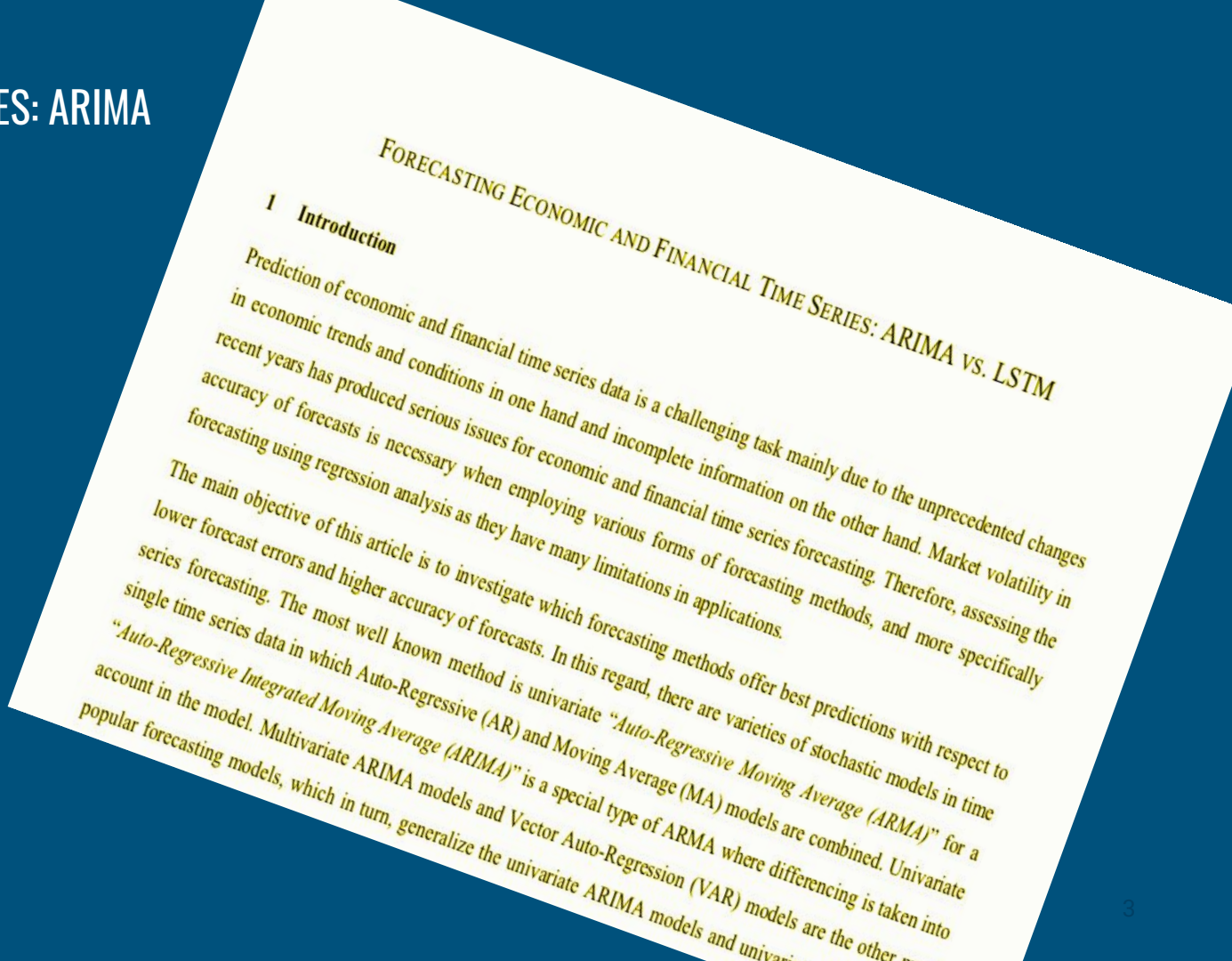
Overview





Deep learning-based approaches have gained popularities among researchers

FORECASTING TIME SERIES: ARIMA VS. LSTM



FORECASTING ECONOMIC AND FINANCIAL TIME SERIES: ARIMA VS. LSTM

1 Introduction

Prediction of economic and financial time series data is a challenging task mainly due to the unprecedented changes in economic trends and conditions in one hand and incomplete information on the other hand. Market volatility in recent years has produced serious issues for economic and financial time series forecasting. Therefore, assessing the accuracy of forecasts is necessary when employing various forms of forecasting methods, and more specifically forecasting using regression analysis as they have many limitations in applications.

The main objective of this article is to investigate which forecasting methods offer best predictions with respect to lower forecast errors and higher accuracy of forecasts. In this regard, there are varieties of stochastic models in time series forecasting. The most well known method is univariate *"Auto-Regressive Moving Average (ARMA)"* for a single time series data in which Auto-Regressive (AR) and Moving Average (MA) models are combined. Univariate *"Auto-Regressive Integrated Moving Average (ARIMA)"* is a special type of ARMA where differencing is taken into account in the model. Multivariate ARIMA models and Vector Auto-Regression (VAR) models are the other popular forecasting models, which in turn, generalize the univariate ARIMA models and univari

FORECASTING TIME SERIES: ARIMA VS. LSTM

Table 2. The RMSEs of ARIMA and LSTM models.

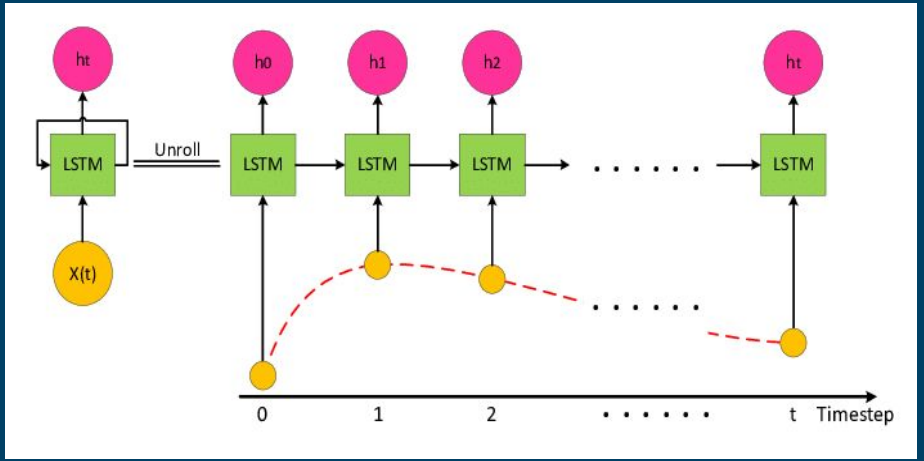
Financial Data	RMSE		% Reduction in RMSE	Economic Data	RMSE		% Reduction in RMSE
	ARIMA	LSTM			ARIMA	LSTM	
N225	766.45	105.315	-86.259	MC	0.81	0.801	-1.111
IXIC	135.607	22.211	-83.621	HO	0.522	0.43	-17.624
HSI	1,306.954	141.686	-89.159	ER	1.286	0.251	-80.482
GSPC	55.3	7.814	-85.869	FB	0.478	0.397	-16.945
DJI[1]	516.979	77.643	84.981	MS	30.231	3.17	-89.514
DJI[2]	287.6	30.61	-89.356	TR	2.672	0.569	-78.705
Average	511.481	64.213	-87.445	Average	5.999	0.936	-84.394

[1]: Monthly; [2]: Weekly

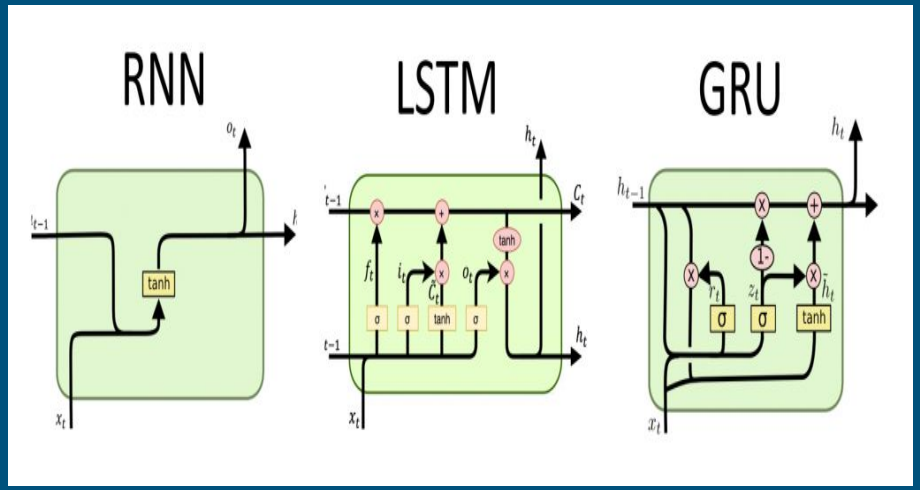
Time series Analysis using LSTM

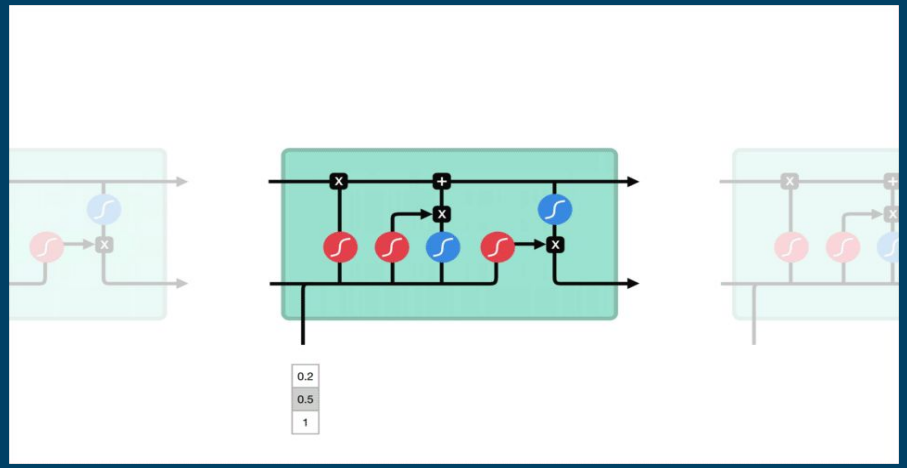
Air Pollution Forecasting:

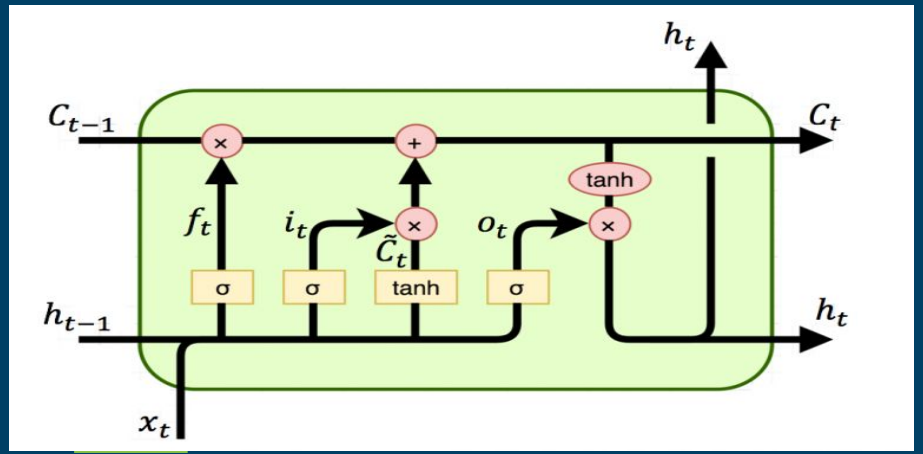
1. Network selection
2. Data preprocessing
3. Network training
4. Testing the model
5. Performance matric (Root-Mean-Square Error (RMSE)) for model evaluation



RNN, LSTM & GRU


RNN
LSTM
GRU
tanh
σ
1-


0.2
0.5
1

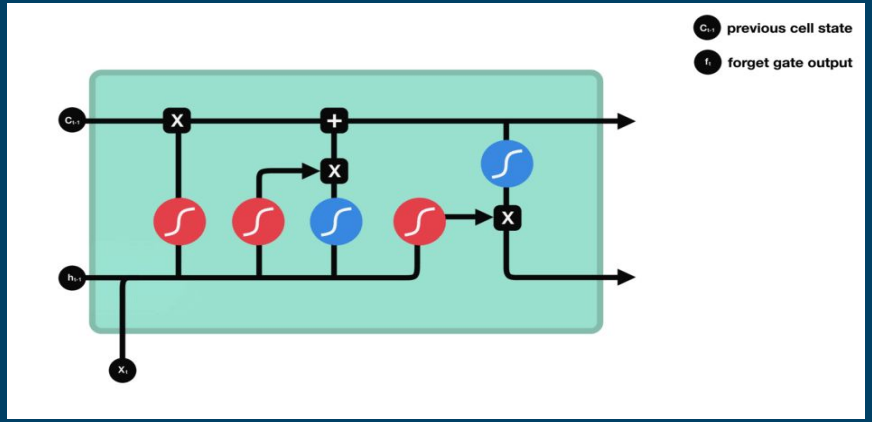

h_t
C_{t-1}
C_t
tanh
f_t
i_t
$\tilde{C}_t$
o_t
σ
σ
tanh
σ
h_{t-1}
h_t
x_t

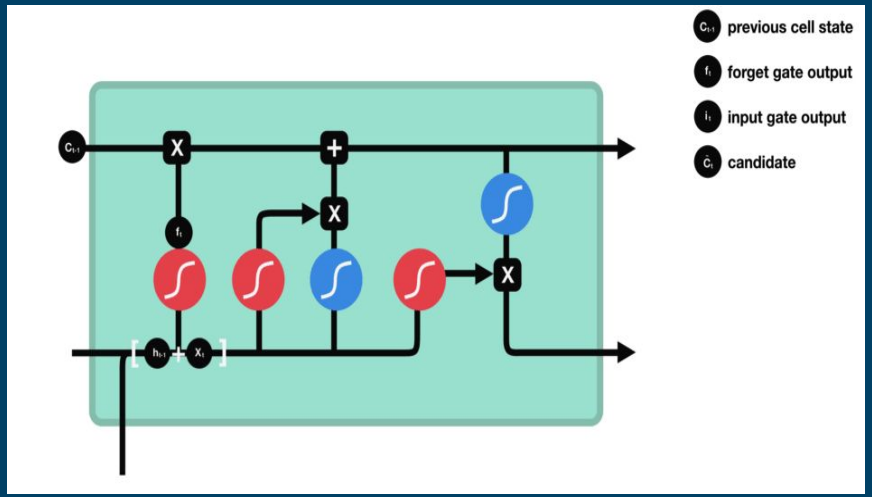
How LSTM Works

Forget gate:
This gate decides what information should be thrown away or kept.

Input gate:

- First, it passes the previous hidden state and current input into a sigmoid function.
- then pass the hidden state and current input into the tanh function to squish values between -1 and 1 to help regulate the network
- Then multiply the tanh output with the sigmoid output
- The sigmoid output will decide which information is important to keep from the tanh output.





How LSTM Works

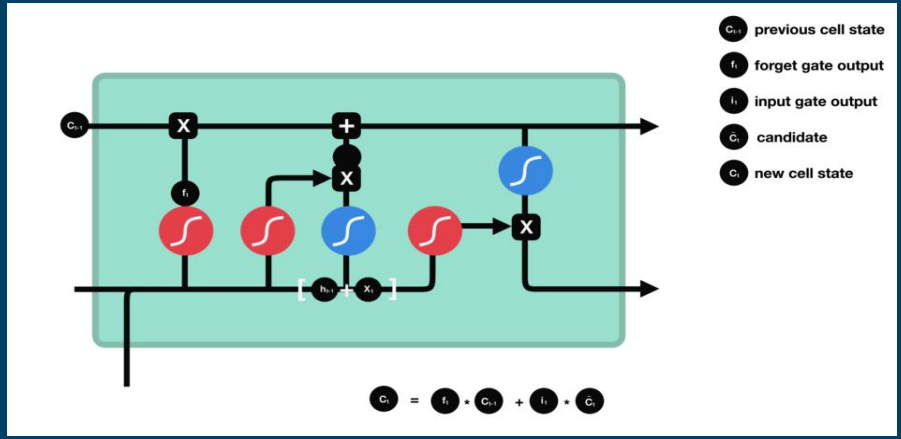
Cell State:

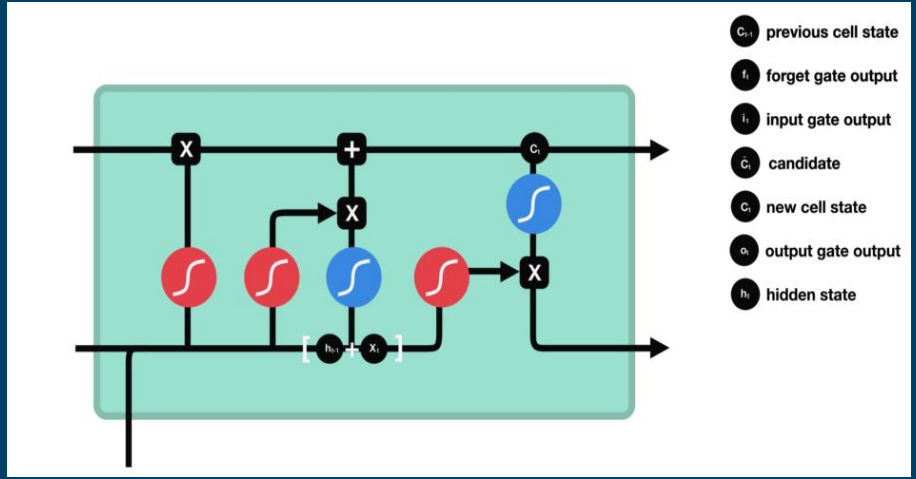
- First, the cell state gets pointwise multiplied by the forget vector
- Then takes the output from the input gate and do a pointwise addition which updates the cell state to new values that the neural network finds relevant. That gives us new cell state

Output Gate:

- First, we pass the previous hidden state and the current input into a sigmoid function.
- Then we pass the newly modified cell state to the tanh function.
- We multiply the tanh output with the sigmoid output to decide what information the hidden state should carry.
- The output is the hidden state. The new cell state and the new hidden is then carried over to the next time step.

Ref:https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21





How GRU Works

Update Gate

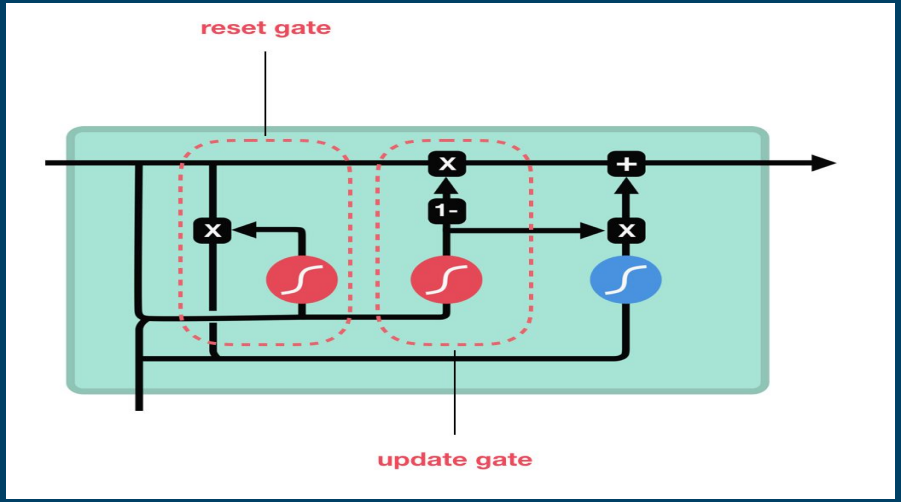
The update gate acts similar to the forget and input gate of an LSTM. It decides what information to throw away and what new information to add

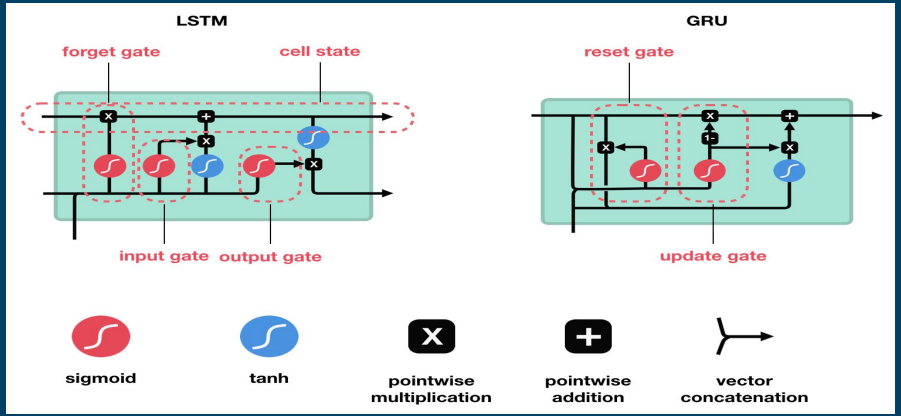
Reset Gate

The reset gate is another gate is used to decide how much past information to forget.

GRU's has fewer tensor operations than LSTM

Ref:https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21





Dataset Preprocessing

1. Data cleaning
2. Use heatmap to see correlation between variables
3. plot each column
4. All features are normalized, then the dataset is transformed into a supervised learning problem

Attribute Information: Multivariate, Time-Series

No: row number

year: year of data in this row

month: month of data in this row

day: day of data in this row

hour: hour of data in this row

pm2.5: PM2.5 concentration (ug/m^3)

DEWP: Dew Point (â„ƒ)

TEMP: Temperature (â„ƒ)

PRES: Pressure (hPa)

cbwd: Combined wind direction

Iws: Cumulated wind speed (m/s)

Is: Cumulated hours of snow

Ir: Cumulated hours of rain

```
In [169]: # manually specify column names
dataset.columns = ['pollution', 'dew', 'temp', 'press', 'wnd_dir', 'wnd_spd', 'snow', 'rain']
dataset.index.name = 'date'
# mark all NA values with 0
dataset['pollution'].fillna(0, inplace=True)
# drop the first 24 hours
dataset = dataset[24:]
# summarize first 5 rows
print(dataset.head(5))
# save to file
dataset.to_csv('pollution.csv')
```

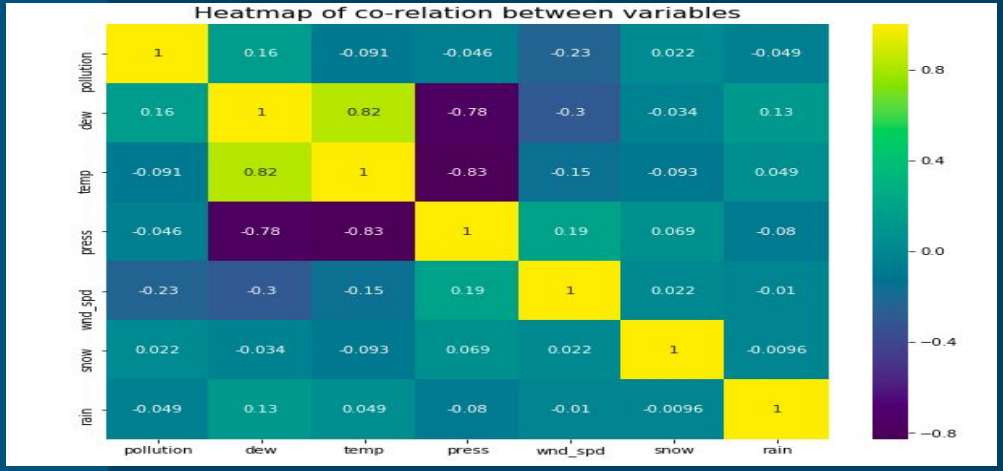
date	pollution	dew	temp	press	wnd_dir	wnd_spd	snow	rain
2010-01-02 00:00:00	129.0	-16	-4.0	1020.0	SE	1.79	0	0
2010-01-02 01:00:00	148.0	-15	-4.0	1020.0	SE	2.68	0	0
2010-01-02 02:00:00	159.0	-11	-5.0	1021.0	SE	3.57	0	0
2010-01-02 03:00:00	181.0	-7	-5.0	1022.0	SE	5.36	1	0
2010-01-02 04:00:00	138.0	-7	-5.0	1022.0	SE	6.25	2	0

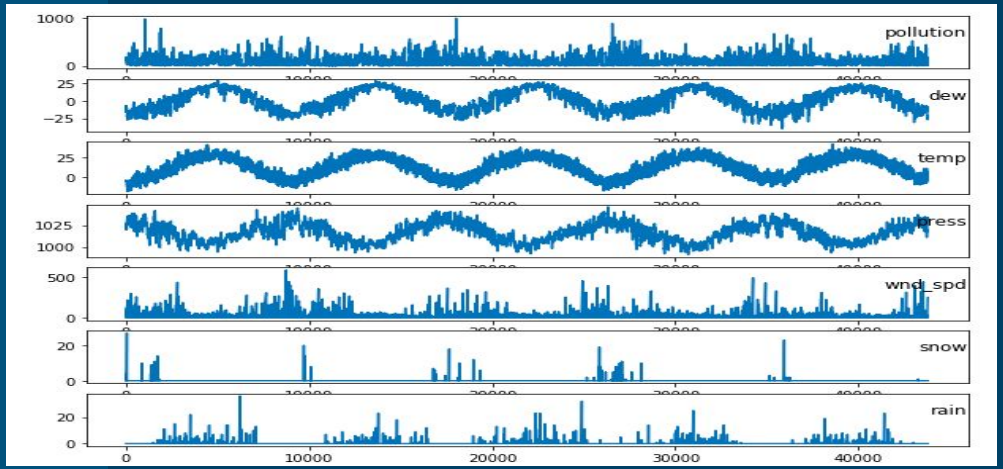
Dataset Preprocessing

Attribute Information: Multivariate, Time-Series

info():

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 43800 entries, 2010-01-02 00:00:00 to 2014-12-31 23:00:00
Data columns (total 8 columns):
pollution 43800 non-null float64
dew 43800 non-null int64
temp 43800 non-null float64
press 43800 non-null float64
wnd_dir 43800 non-null object
wnd_spd 43800 non-null float64
snow 43800 non-null int64
rain 43800 non-null int64
dtypes: float64(4), int64(3), object(1)
memory usage: 3.0+ MB





Dataset Preprocessing

1. We have used 3 hours of data as input
2. We have 3 * 8 + 8 columns in our framed dataset
3. We have taken 3 * 8 or 24 columns as input for the obs of all features across the previous 3 hours.
4. We have taken just the pollution variable as output at the following hour

	var1(t-3)	var1(t-2)	var1(t-1)	var1(t)
3	0.12977867	0.14889336	0.15995975	0.18209255
4	0.14889336	0.15995975	0.18209255	0.13883299
5	0.15995975	0.18209255	0.13883299	0.10965794
6	0.18209255	0.13883299	0.10965794	0.1056338
7	0.13883299	0.10965794	0.1056338	0.12474848
8	0.10965794	0.1056338	0.12474848	0.12072434
9	0.1056338	0.12474848	0.12072434	0.13279678
10	0.12474848	0.12072434	0.13279678	0.14084506
11	0.12072434	0.13279678	0.14084506	0.1529175
12	0.13279678	0.14084506	0.1529175	0.14889336
13	0.14084506	0.1529175	0.14889336	0.16498993
14	0.1529175	0.14889336	0.16498993	0.15895371
15	0.14889336	0.16498993	0.15895371	0.15492958
16	0.16498993	0.15895371	0.15492958	0.15995975
17	0.15895371	0.15492958	0.15995975	0.16498993
18	0.15492958	0.15995975	0.16498993	0.17102616
19	0.15995975	0.16498993	0.17102616	0.1498994
20	0.16498993	0.17102616	0.1498994	0.15492958

Design network & Result

LSTM with and without Dropout layer:

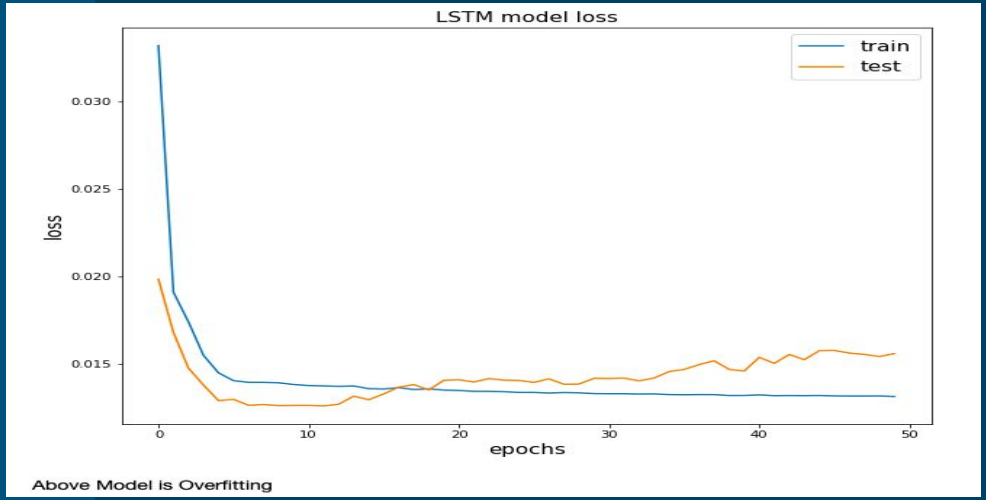
Total train data was for first 4yrs. Testing data was for last 1yr

Loss function: mean squared error

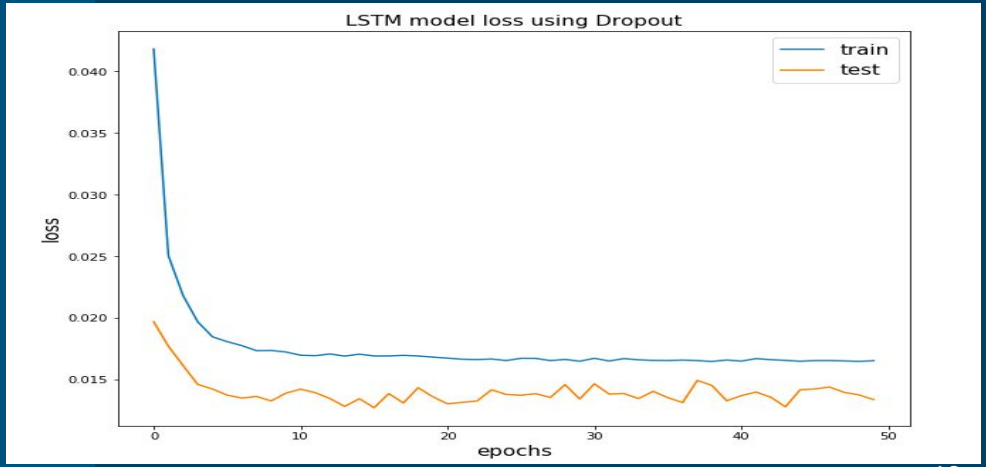
Optimization algorithm: ADAM

Batch Size and Iterations: 72

Epoch: 50



Above Model is Overfitting



Design network & Result

GRU

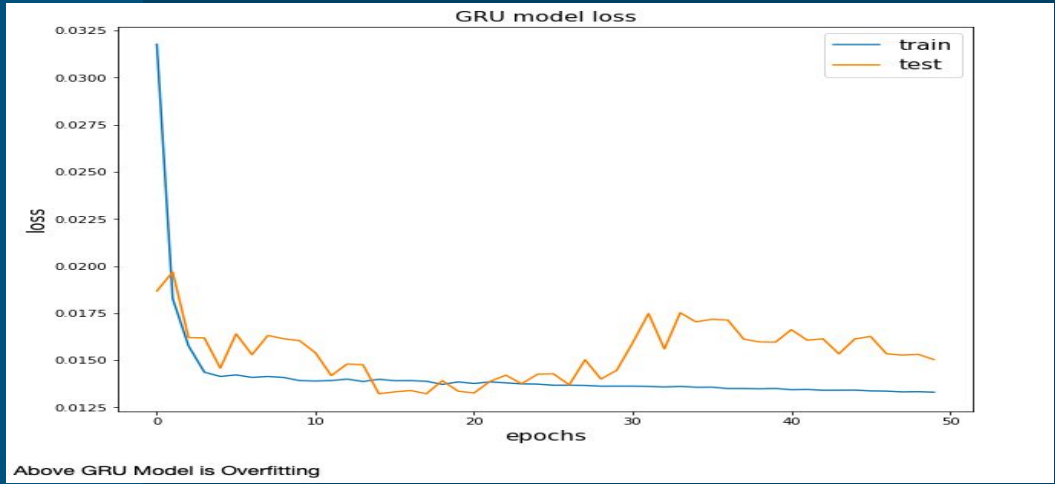
Total train data was for first 4yrs. Testing data was for last 1yr

Loss function: mean squared error

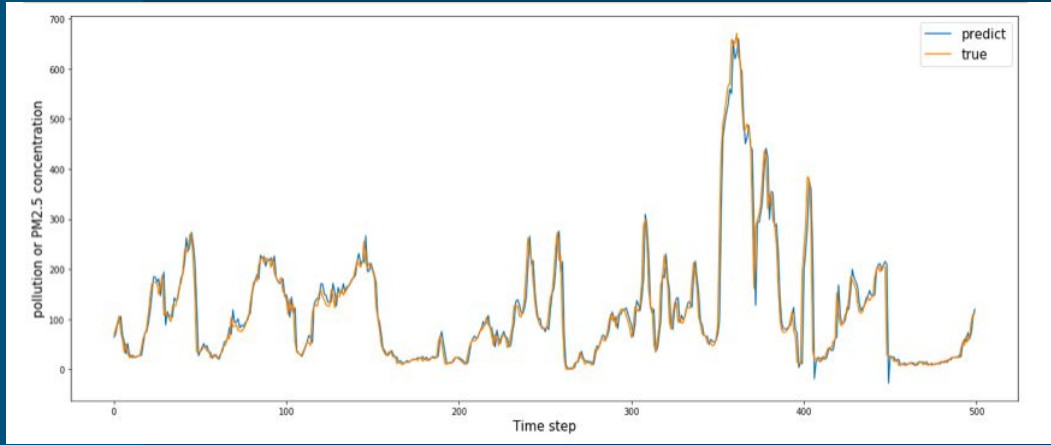
Optimization algorithm: ADAM

Batch Size and Iterations: 72

Epoch: 50



Above GRU Model is Overfitting



Design network & Result

Overcome overfitting and underfitting:

- Cross-validation
- Train with more/less data
- Remove features
- Early stopping
- Dropout Regularization

Total train data was for first 1yrs. Testing data was for last 4yrs

Test RMSE FOR LSTM:: 26.349