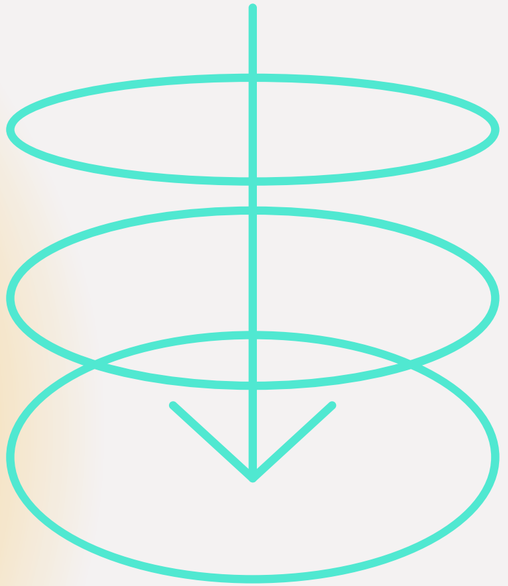
# An Optimized Deep Learning Approach for Forecasting Temperature

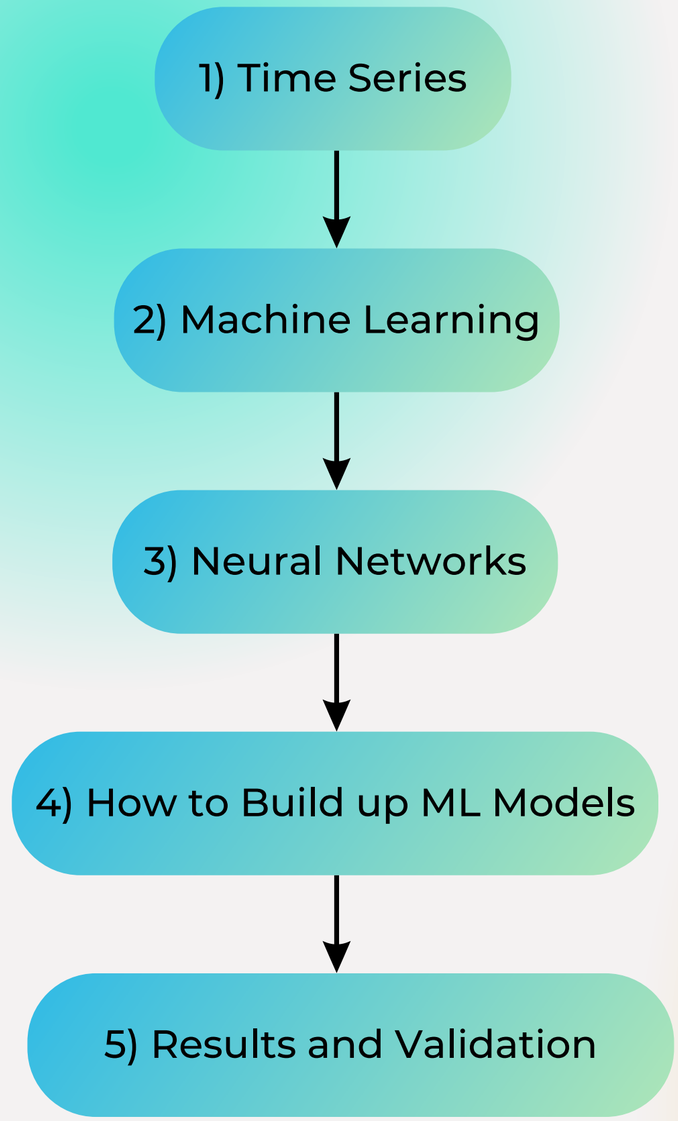


Prof.Dr. Şeref Naci Engin

Çağatay Berke BOZLAK – 21567009

Supervisor: Asst.Prof.Dr. Claudia F. Yaşar

# CONTEXT


1) Time Series
2) Machine Learning
3) Neural Networks
4) How to Build up ML Models
5) Results and Validation

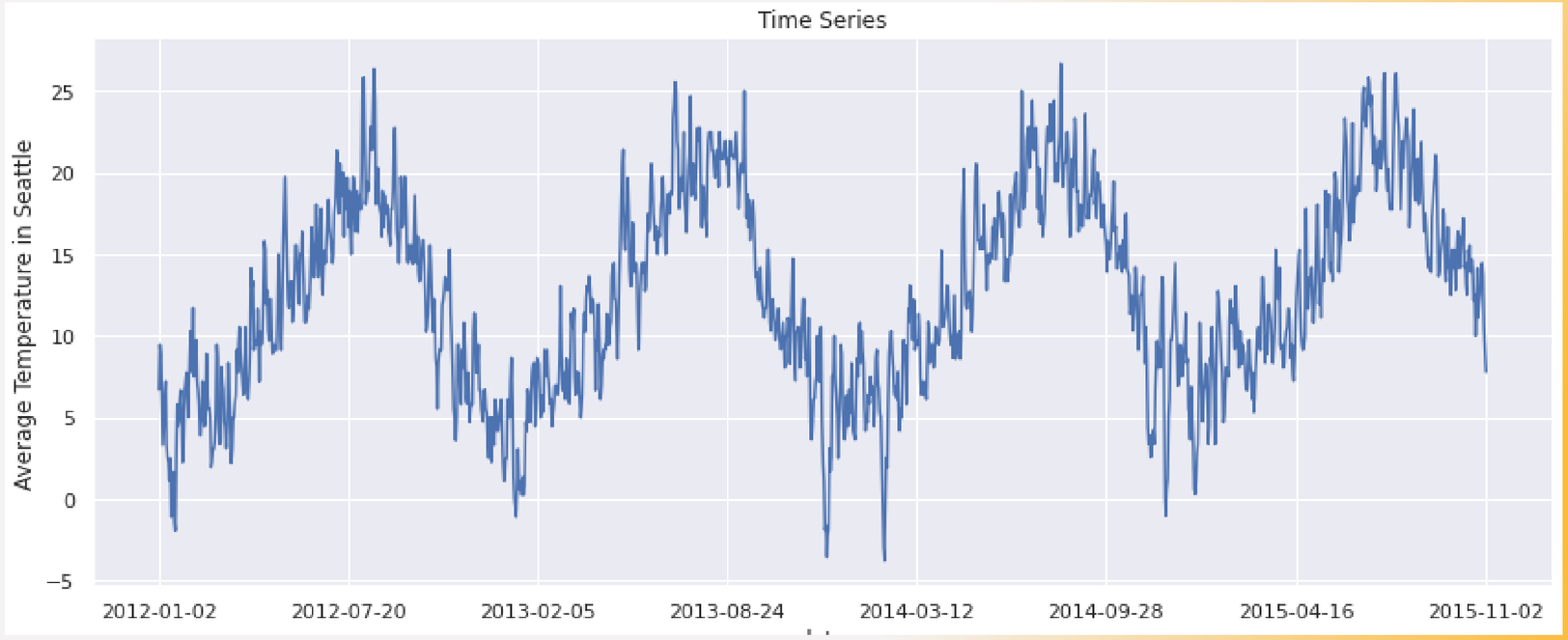
# Time Series

## What is it?

Time series refers to a chain of data points observed and recorded in a time order over a specific period. It represents the output obtained from monitoring and tracking specific events or processes.

## Why is it important to forecast time series?

Time series forecasting provides valuable insights that enable organizations to optimize operations, improve efficiency, and make informed decisions for better performance and profitability.



# Machine Learning

# Machine Learning

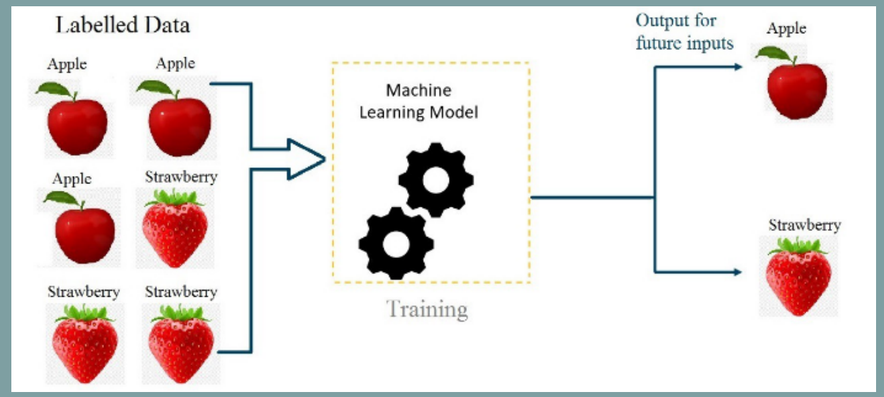
Machine learning is a field of study that focuses on developing algorithms and models that enable computers to learn and make predictions or decisions without being explicitly programmed.

# Deep Learning

Deep learning is a subset of machine learning that utilizes neural networks with multiple layers to learn and extract complex patterns from data, enabling accurate predictions and classifications in various domains such as image recognition, speech processing, and natural language understanding.

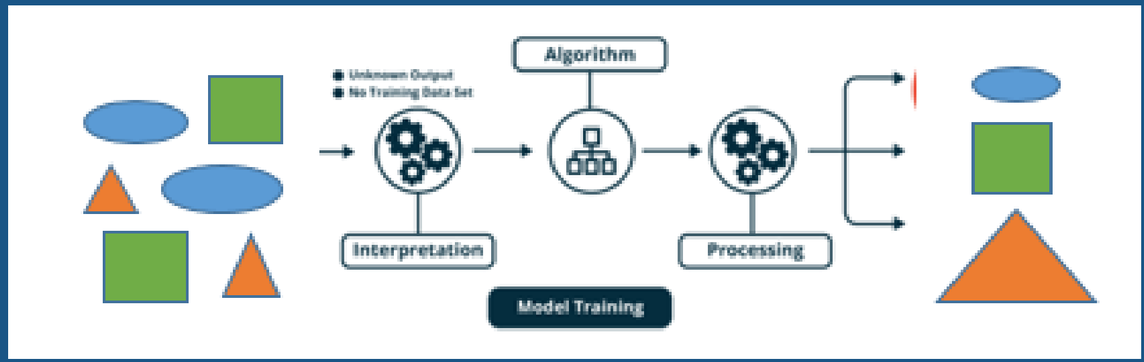
## Supervised Learning

Supervised learning uses labeled data to train a model that can make predictions or classifications on new data by learning from the provided labels.




## Unsupervised Learning

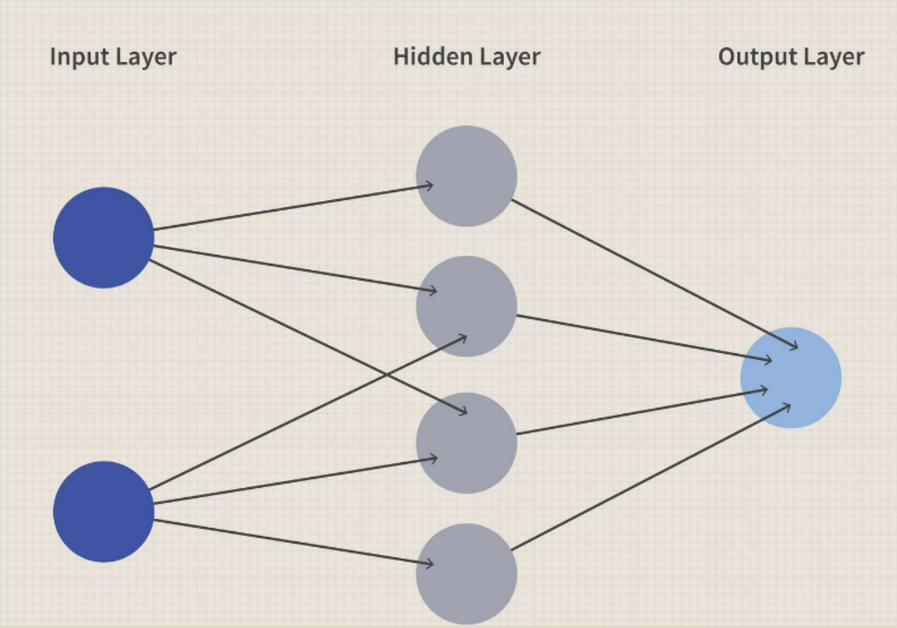
Unsupervised learning explores unlabeled data to find patterns and structures without explicit guidance or predefined labels.



# Neural Networks

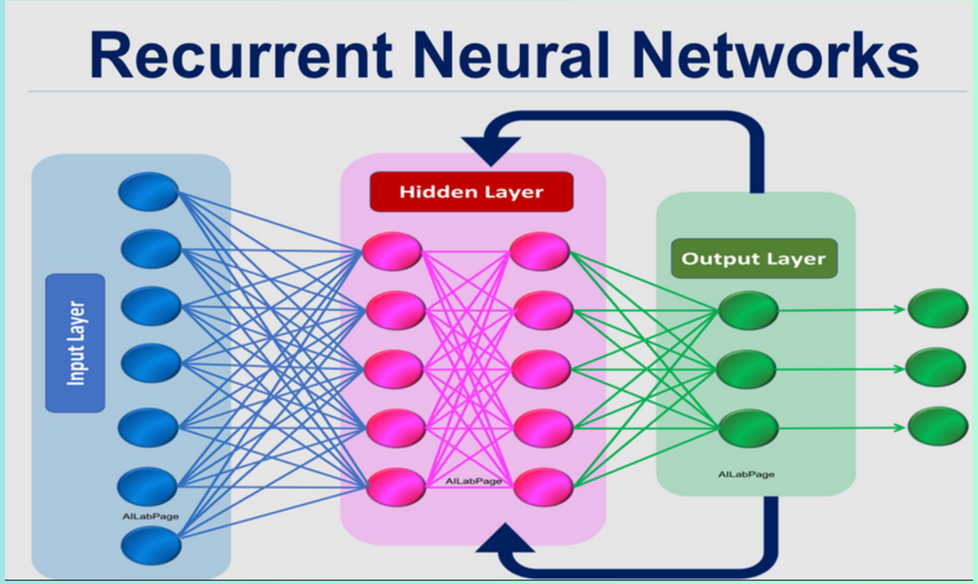
# Neural Networks

Neural networks are computational models inspired by the brain's structure. They consist of interconnected artificial neurons organized into layers. Neural networks learn from data by adjusting connection weights through training. They excel in tasks like image recognition, speech processing, and natural language understanding.



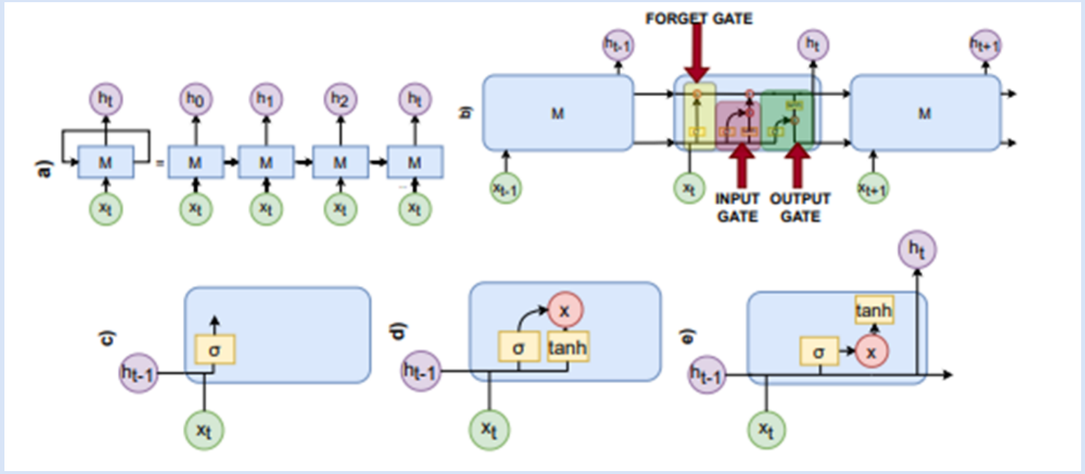

# Reccurent Neural Networks

Recurrent Neural Networks (RNNs) process sequential data by retaining information from previous inputs. They excel in tasks like natural language processing, speech recognition, and time series analysis. RNNs capture dependencies within sequences and have wide-ranging applications in understanding and generating sequential data.



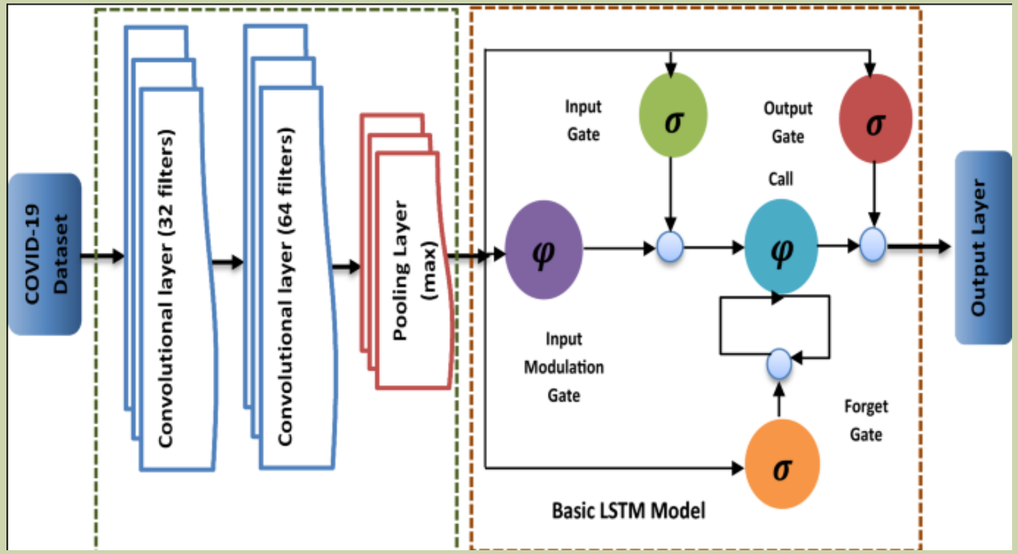
# LSTM Models

LSTM (Long Short-Term Memory) models are a type of recurrent neural network designed to capture long-term dependencies in sequential data. They overcome the vanishing gradient problem and excel in tasks like time series analysis, natural language processing, and speech recognition. LSTMs utilize memory cells and gates to retain and regulate information, making them effective for modeling complex sequential patterns.

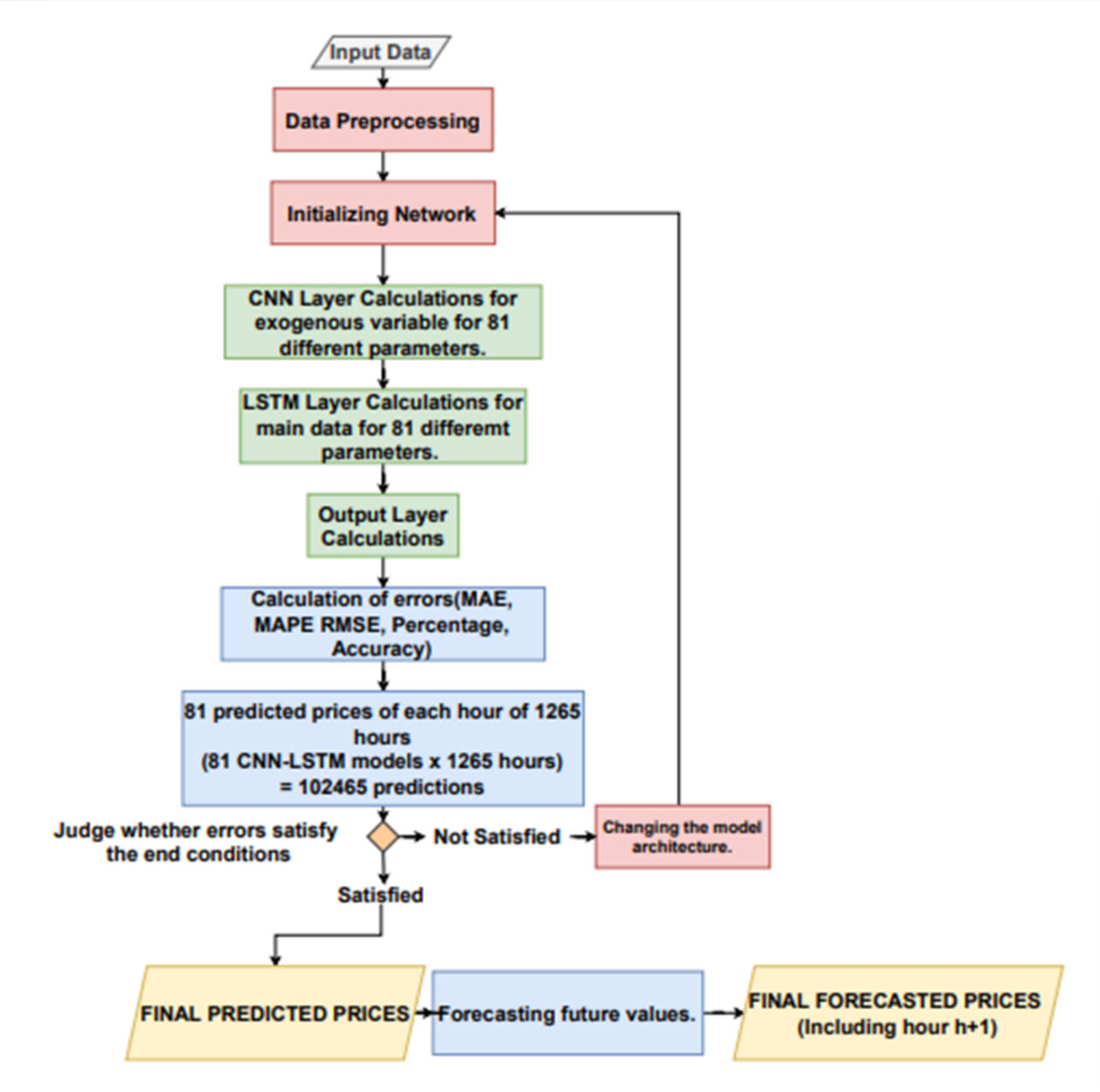



# CNN-LSTM Models

A CNN-LSTM model is a combination of CNN layers that extract the feature from input data and LSTMs layers to provide sequence prediction. The CNN-LSTM is generally used for having an exogenous variable for predictions with LSTMs



# How to Build up ML Models



# ENVIRONMENT

## Python

is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis.

## Google Colaboratory

is a freemium tool offered by Google Research that allows users to write and exe-cute Python code in their web browsers.

## TensorFlow

platform helps you implement best practices for data automation, model tracking, performance monitoring, and model retraining. The most useful advantage of TensorFlow is that it lets you use the GPU instead of the CPU.

## Keras

is a high-level, deep learning API developed by Google for implementing neural networks. It is written in Python and is used to make the implementation of neural networks easy.

# Exogenous Variable

Exogenous variables in machine learning are external predictors that help improve predictions or understanding of the target variable by providing additional context and information. In my work I chose humidity as exogenous variable to temperature in CNN-LSTM.

# Importing and Manipulating the Dataset

The date needs to be exported and manipulated in Python. It can be handled with below code snippets.

```
df = pd.read_csv('seattle-weather.csv')
```

```
df_max = df['temp_max']
df_min = df['temp_min']
df['temp_average'] = df['usd/tl']
df
```

In the end manipulated dataset looks like below:

| | date | precipitation | temp_max | temp_min | wind | weather | temp_average |
|---|---|---|---|---|---|---|---|
| 1 | 2012-01-02 | 10.9000 | 10.6000 | 2.8000 | 4.5000 | rain | 6.7000 |
| 2 | 2012-01-03 | 0.8000 | 11.7000 | 7.2000 | 2.3000 | rain | 9.4500 |
| 3 | 2012-01-04 | 20.3000 | 12.2000 | 5.6000 | 4.7000 | rain | 8.9000 |
| 4 | 2012-01-05 | 1.3000 | 8.9000 | 2.8000 | 6.1000 | rain | 5.8500 |
| 5 | 2012-01-06 | 2.5000 | 4.4000 | 2.2000 | 2.2000 | rain | 3.3000 |
| ... | ... | ... | ... | ... | ... | ... | ... |

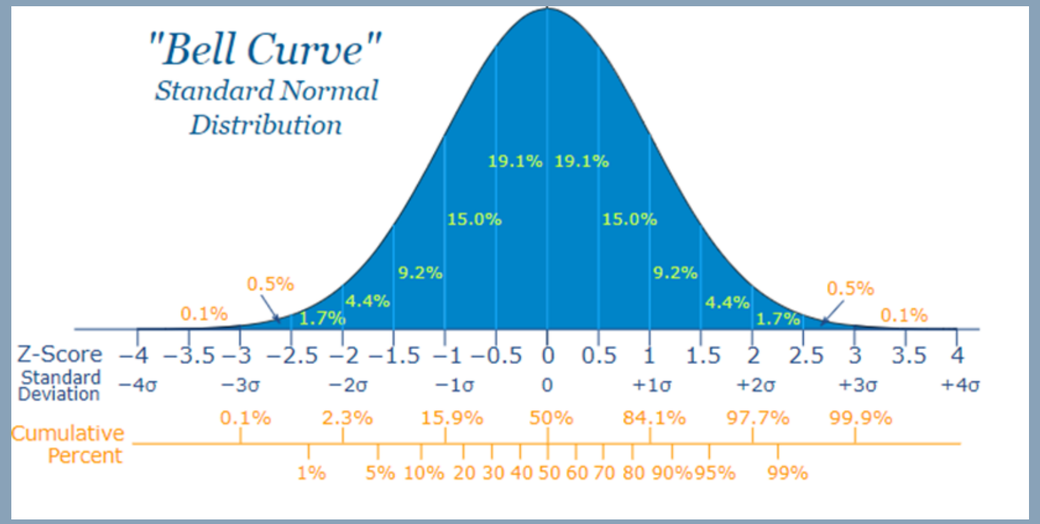
Dataset Check

# Missing Cell Check

We need to be sure that our dataset does not have any missing cells. To check whether the dataset has missing cells .dropna command can be used. If the dataset has missing cells it is needed to be filled using either backward filling or forward filling methods.

# Gaussian Distribution Checks

A distribution is simply a collection of the value and frequency of a given observation, like the age of a population. Samples of an ideal Gaussian distribution (AKA normal distribution or bell curve) follow the bell curve, meaning values are more likely to be around the mean than at the extremes.



## Why is Gaussian Distribution Important in ML

In machine learning, cost functions or neuron potential values are the quantities that are expected to be the sum of many independent processes (such as input features or activation potential of the last layer) and often have distributions that are nearly normal. Knowing the gaussian nature of the dataset, one can continue to use parametric statistics.
Some ML models need to be converted into Gaussian distributions with some methods, such as power transforms, differential transforms, and so on.

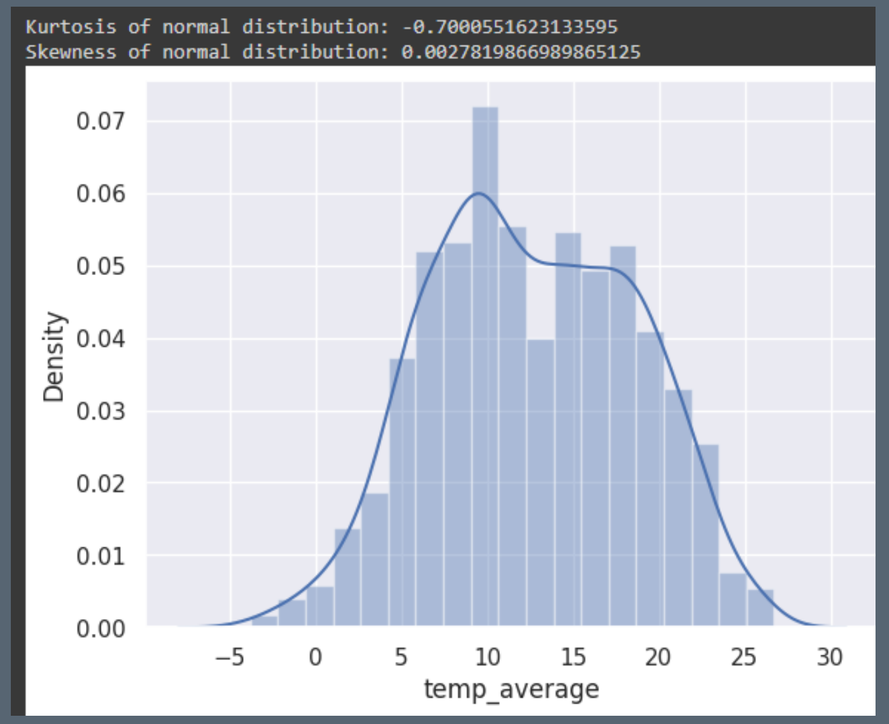
Test Methods

## Skewness

Skewness is usually described as a measure of a dataset's symmetry – or lack of symmetry. A perfectly symmetrical data set will have a skewness of 0. The normal distribution has a skewness of 0.

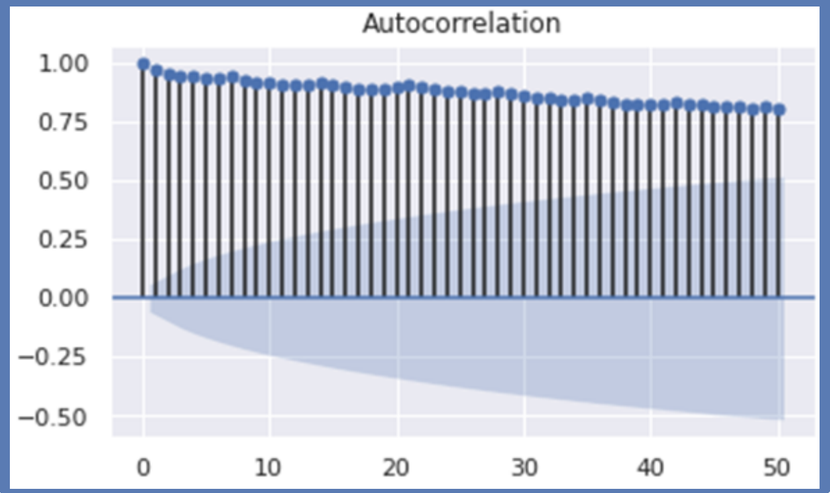
## Kurtosis

Kurtosis describes the "fatness" of the tails found in probability distributions. The normal distribution has a kurtosis of exactly 3.0.



# Correlation and Autocorrelation

Correlation measures the relationship between two variables. Autocorrelation assesses the relationship between a variable and its own past values. It is useful for analyzing time series data and identifying patterns and dependencies within the data.




## Train and Test Dataset

We have to train a model in order to make our model make predictions. If we don't train our model well enough for predictions, the result will come out bad.
In ML, a usually trained model is separated from 80% of the actual data. We want our model to find and learn the hidden parameters inside this model, implement what it learned onto the test dataset, and actually predict this test dataset.

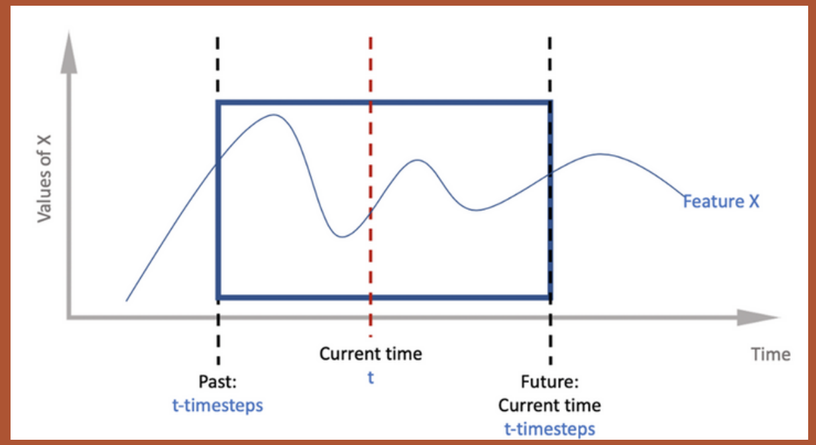

## Scaling a Model

The ML algorithm works better when features are relatively similar in scale and close to normal distribution.
Min-Max scaling is a normalization technique that enables us to scale data in a dataset to a specific range (0–1 in our case) using each feature's minimum and maximum value.
Unlike standard scaling, where data are scaled based on the standard normal distribution (with mean = 0 and standard deviation = 1), the min-max scaler uses each column's minimum and maximum value to scale the data series.
The scale of data for some features may be significantly different from those of others, which may harm the performance of our models. It is especially the case with algorithms that rely on a measure of distance, such as neural networks and KNN.

```
scaler = MinMaxScaler(feature_range=(0, 1))
```

## Time Steps

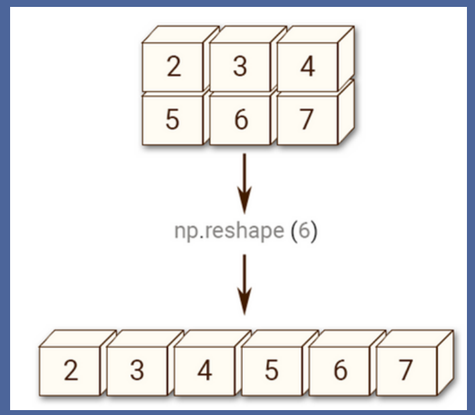
Time steps are a way of examining and analyzing your data through specified time intervals.
By setting the time steps, we define how many steps back our model should look for hidden patterns. We can forecast future steps in time.




Reshaping and Hidden Layers

## Reshaping

Reshaping means changing the shape of an array. The shape of an array is the number of elements in each dimension. By reshaping we can add or remove dimensions or change number of elements in each dimension.




## Hidden Layers

Hidden Layers detect the neuron numbers that our models will use. There is a common formula to obtain hidden layers.

Hidden Layers=(Input Numbers)/((α x (Output Numbers+Time Steps)))

where α is between 2 and 10

```
X_test.shape
realvalues = Y_test
hidden_layers = int(X_test.shape[0]/(a*(X_test.shape[1]+X_test.shape[2])))
```

- Train-test split ratio
- Learning rate in optimization algorithms (e.g. gradient descent)
- Choice of optimization algorithm (e.g., gradient descent, stochastic gradient descent, or Adam optimizer)
- Choice of activation function in a neural network (nn) layer (e.g. Sigmoid, ReLU, Tanh)
- The choice of cost or loss function the model will use
- Number of hidden layers in a nn
- Number of activation units in each layer
- The drop-out rate in nn (dropout probability)
- Number of iterations (epochs) in training a nn
- Number of clusters in a clustering task
- Kernel or filter size in convolutional layers
- Pooling size
- Batch size

# Hyperparameters

Hyperparameters are parameters whose values control the learning process and determine the values of model parameters that a learning algorithm ends up learning. The prefix ‘hyper_’ suggests that they are ‘top-level’ parameters that control the learning process and the model parameters that result from it.

Hyperparameters of LSTM Model creates the architecture of the model. Belowthe model architecture that I’ve used can be seen below:

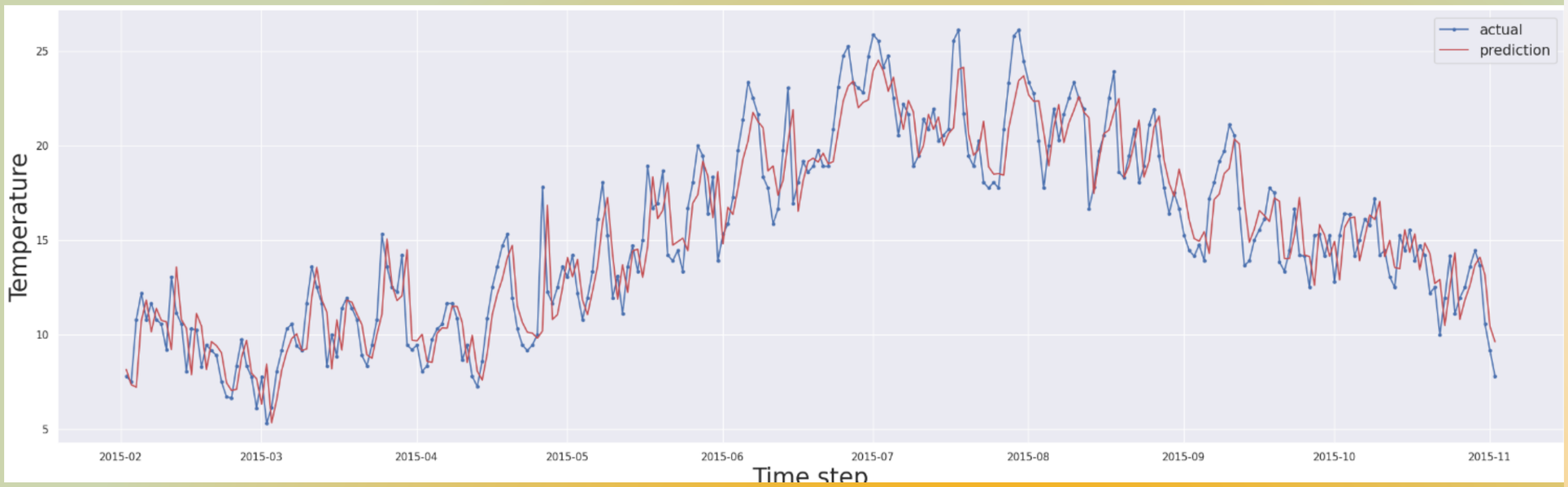
```
model = Sequential()
model.add((LSTM(hidden_layers, return_sequences=False , activation = 'relu',recurrent_activation='tanh',
               kernel_initializer='orthogonal',recurrent_initializer='orthogonal',input_shape=(X_test.shape[1],X_test.shape[2]))))
model.add(Dense(hidden_layers*2/3,activation='relu'))
model.add(Dense(1))
model.compile(loss='mse', optimizer='Adam')

history = model.fit(X_train, Y_train, epochs=300, batch_size= 32,validation_data=(X_test, Y_test),
                    callbacks=[EarlyStopping(monitor='val_loss', patience=10)], verbose=1, shuffle=False)
# Training Phase
```

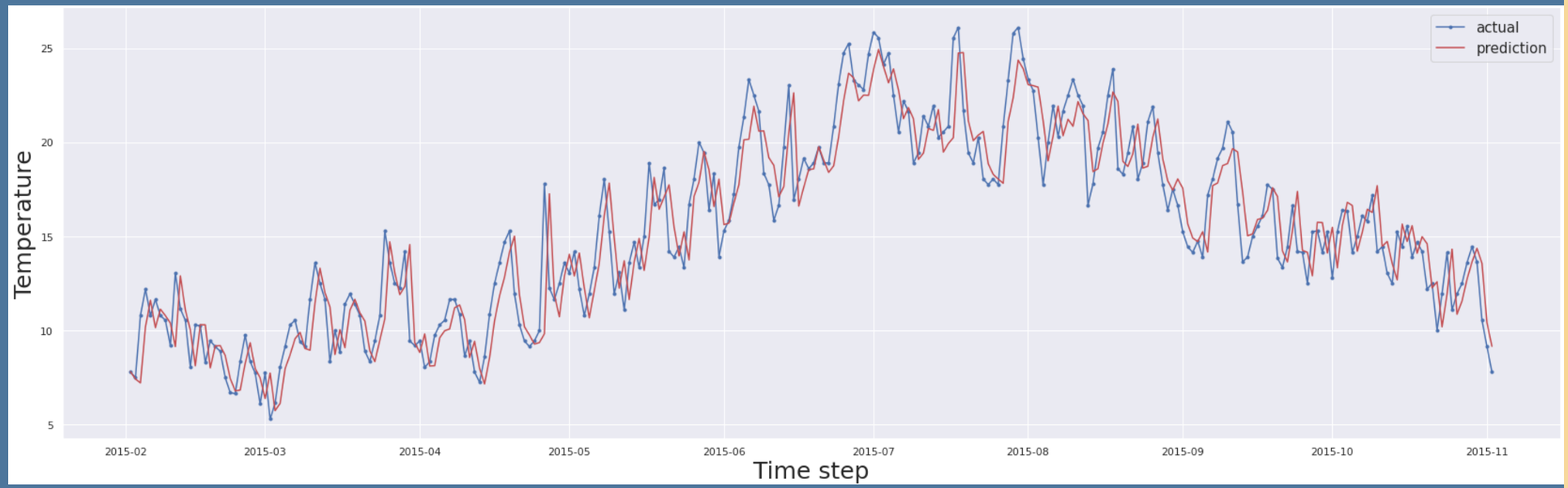
# Results and Validation

# PREDICTION PLOTS

LSTM

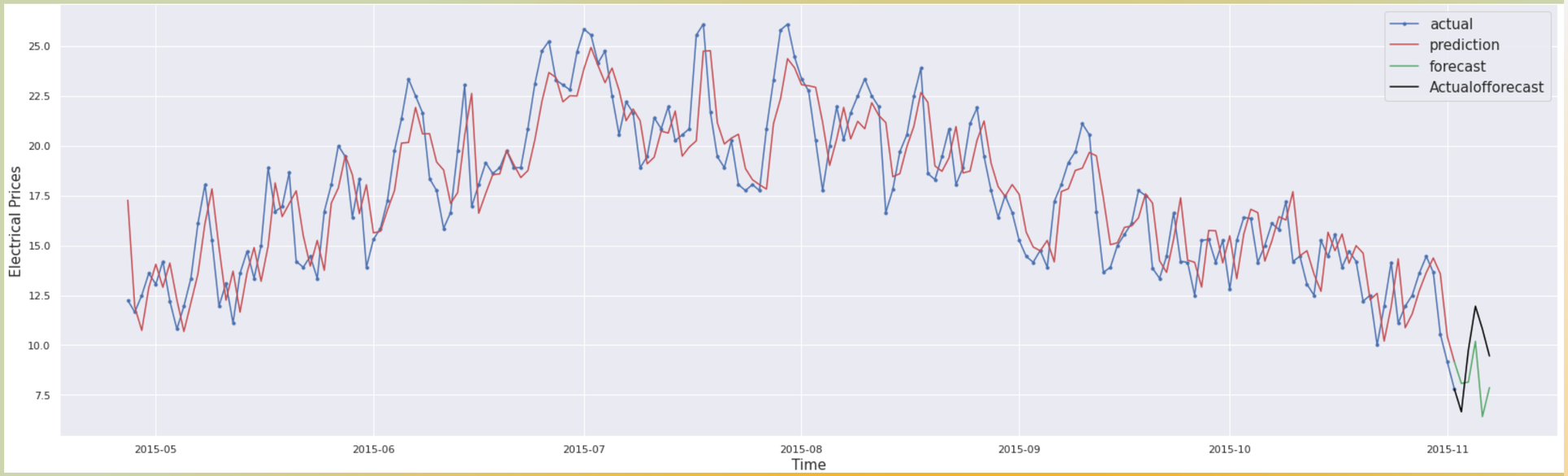



CNN-LSTM

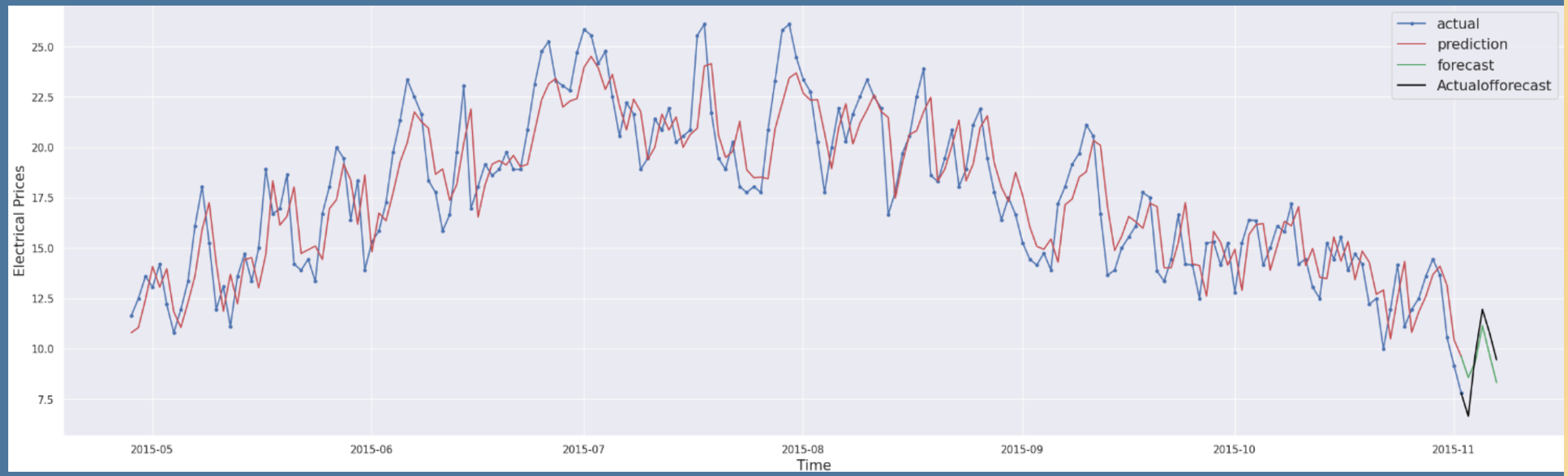


# FORECAST PLOTS

LSTM




CNN-LSTM



# Validation

CNN–LSTM performed better, as evidenced by the plot and error table. It is because the analysis of exogenous variable of CNN model assisted the LSTM model in better analyzing the input and this results indicates that the exogenous variable selection was correct.
In the error table, we can clearly see why we need to split our dataset into train and test data; the test validation errors are always lower.

| Validation Error | CNN-LSTM | LSTM |
|---|---|---|
| Train Mean Absolute Error | 1.4377 | 1.4874 |
| Train Root Mean Squared Error | 1.7967 | 1.8674 |
| Test Mean Absolute Error | 1.4596 | 1.4698 |
| Test Root Mean Squared Error | 1.8300 | 1.8630 |

# Thanks