Deep Neural Network for Weather Time Series Forecasting

Contents

1. Abstract

Forecasting future values of a time series plays an important role in nearly all fields of science and engineering, such as economics, finance, business intelligence and industrial applications, also in real world applications such as speech recognition, real time sign language translation, finance markets, weather forecast etc. Deep Learning algorithms are known to perform best when there is a massive dataset available for learning. Not every time series problem has massive dataset available. In that case advanced ML algorithms are available for time series applications. Because of the nature of time series analysis problem where two values of the same feature in two different time steps are considered as different features, the data size available for processing becomes larger. It is hard to decide which algorithms will perform better for a medium size dataset. The recommendation made in this project can be useful for anyone looking for best Machine Learning/Deep Learning algorithms for medium size time series problem dataset. I used Historical hourly Weather Data from Kaggle website. The dataset contains ~5 years of high temporal resolution (hourly measurements) data of various weather attributes, such as temperature, humidity, air pressure, etc. There is non-temporal data such as longitude and latitude of cities, which is also used in forecasting. Both Univariate and Multivariate time series analysis is done to collect the data for the recommendation.

2. Time Series Analysis

A time series is a sequence of data observations collected over a period of time at regular intervals. The temporal dependency in time series cause two otherwise identical points of time to belong to different classes or predict different behavior. This characteristic generally increases the difficulty of analyzing them.

3. Weather Forecast

Weather forecast is among the most popular forecast problems. Weather forecast includes forecasting temperature, pressure, humidity, wind direction and wind speed. Unlike other time series datasets, weather data has unique features. There is season-to-season, year-to-year variability in the trends of weather data. The temperatures and Pressures are correlated. Wind speed and direction have similar attributes and changing patterns. So all these features could be separately predicted using univariate time series analysis techniques or could be used jointly to predict using multivariate time series techniques.

4. Univariate and Multivariate Time series Analysis

A univariate time series is a series with a single time-dependent variable. To predict the temperature for the next few days, we look at the past values and try to extract a pattern. We would notice that the temperature is lower in the morning and at night, also we notice that it is colder during the months of November to January and warmer in April to June. In these observations we used only variable that is temperature. So this is called Univariate Time Series Analysis.

A Multivariate time series has more than one time-dependent variable. Each variable depends not only on its past values but also has some dependency on other variables. This

dependency is used for forecasting future values. For example, the temperature at any time depends on atmospheric pressure, humidity, wind speed, wind direction. To accurately predict the temperature, other variables should be considered too. This is called Multivariate Time Series Analysis.

5. Machine Learning for Time Series Analysis

There has been an extensive research done in machine learning for time series forecasting techniques. Several machine learning algorithms are used to solve time series forecast problems, such as LinearRegression from Scikit-learn library *[13]*, RandomForest from Scikit-learn library *[12]*, XGBRegressor from Xgboost library *[11]*, and Auto Regressive Integrated Moving Average (ARIMA) *[10]* model from statsmodel library. Weather Forecasting requires complex dynamic analysis of data from several past years which has variability from one year to another. Linear models such as LinearRegression, ARIMA cannot capture the non-linearity of the weather time series. Nonlinear Machine Learning models such as RandomForest and XGBRegressor perform better in precisely forecasting weather.

6. Historical Hourly Weather Dataset

The dataset [*1*] contains about five years of high temporal resolution i.e. hourly measurements of six weather attributes: temperature, humidity, air pressure, wind direction, wind speed and general weather description of 30 US and Canadian cities and 6 Israeli cities. The dataset contains separate file for each weather attribute. Each file contains 36 cities as columns. Rows are time axis and time axis are same in all files for different weather attributes. There are 44460 rows for each city making it total of 1660560 samples of each five weather features. The dataset also has non temporal data, the location of each city in term of longitude and latitude, which is also used in forecasting. Both Univariate and Multivariate time series analysis is done to collect the data for the recommendation.

7. Data Wrangling

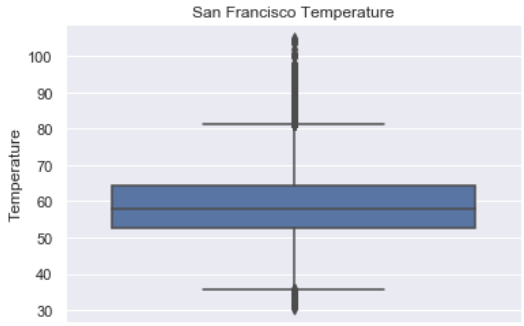
The following Data Wrangling steps are performed:

- There are missing values in the datasets. For several cities first full months' data is missing. Deleted those time steps as filling those values could cause the distraction in the training of the models.
- There are other values missing which were forward fill and then backward fill to make sure all the entries are filled.
- Wind Speed and Wind Direction have a value of 0 for several samples. Replaced them with a small value of 0.001 to avoid divide by zero error later during the evaluation of MAPE (mean absolute percent error) performance metric.
- Temperature, Pressure and Wind Speed are converted into popular units of Degree Fahrenheit, inches of Mercury and miles per hour respectively.
- Cleaned Data is stored in separate csv files for later univariate and multivariate ML and DL analysis.

8. Exploratory Data Analysis

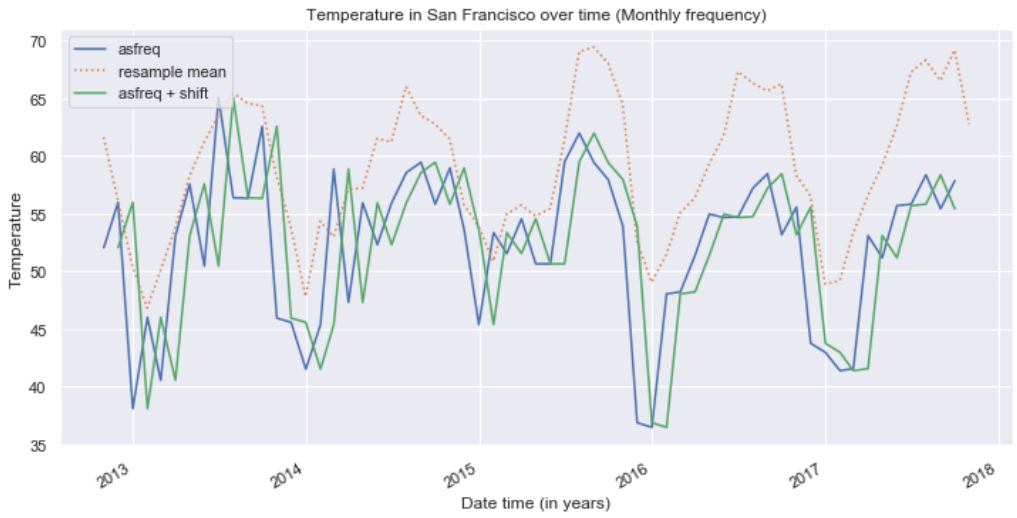
a. Outliers

The Box Plot of San Francisco temperature shows that there are some outliers according to the statistics. I did not try to change anything for these outliers since they represent extreme weather conditions which happen in real world time to time. I modeled the time series with these outliers and see if the model can find a pattern and predict the extreme weather for future.



b. Resampling and Converting Frequencies

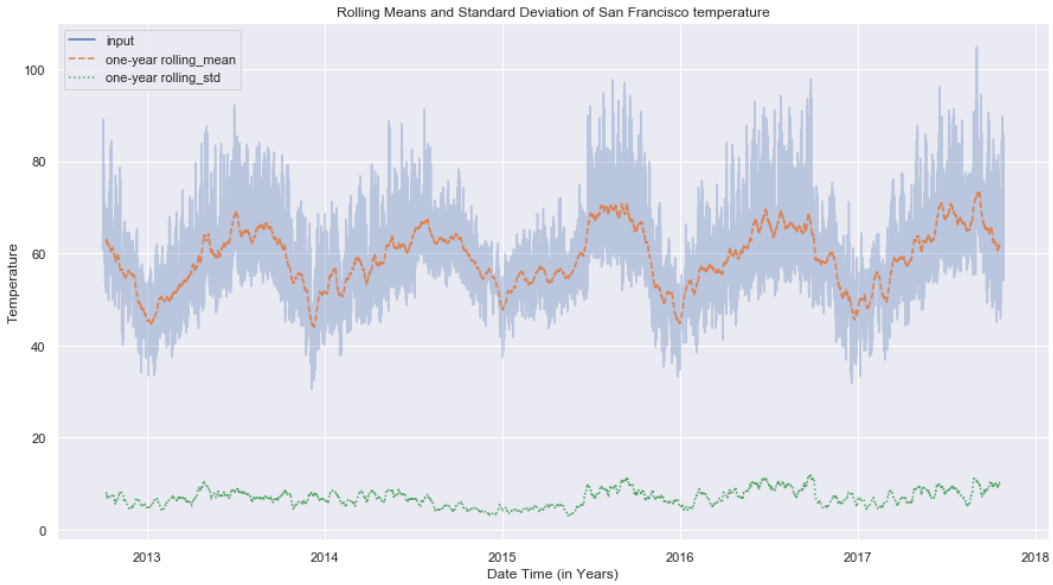
The plot below shows the monthly temperature of San Francisco and the temperature at a lag of one month. The graph consists of straight lines which indicates Temperature is a Linear Time Series so gradient is constant between two local minima and maxima values of the graph. The gradients changes between positive and negative value over the period of time. Negative gradient indicates negative correlation and positive gradient indicates positive relationship between time and values of the Temperature.



c. Rolling windows

Rolling statistics are time series-specific operation implemented by Pandas. The rolling view makes available a number of aggregation operations by default. For example,

here is the one-year centered rolling mean and standard deviation of the San Francisco Temperature: The visuals of the plot show that rolling mean and rolling standard deviation are constant.



d. Deterministic vs Non deterministic Time series

Time series can be deterministic or non-deterministic in nature. Deterministic time series always behave in an expected manner where as non-deterministic time series is stochastic or random in nature. The following measure indicate is a time series is deterministic or not.

Covariance Stationary - If a time series mean, variance and covariance with past and future values do not change over time then the model is known to be covariance stationary. Time series needs to meet following three criteria to be stationary:

1. Constant Mean - Mean or expected value of a time series over successive time periods needs to be constant for a time series to be considered covariance stationary. This implies that the expected value should not be time dependent.

2. Constant Variance - Variance or standard deviation of a time series needs to be constant over time and should not be dependent on time. This is the second criteria for a time series to be covariance stationary.

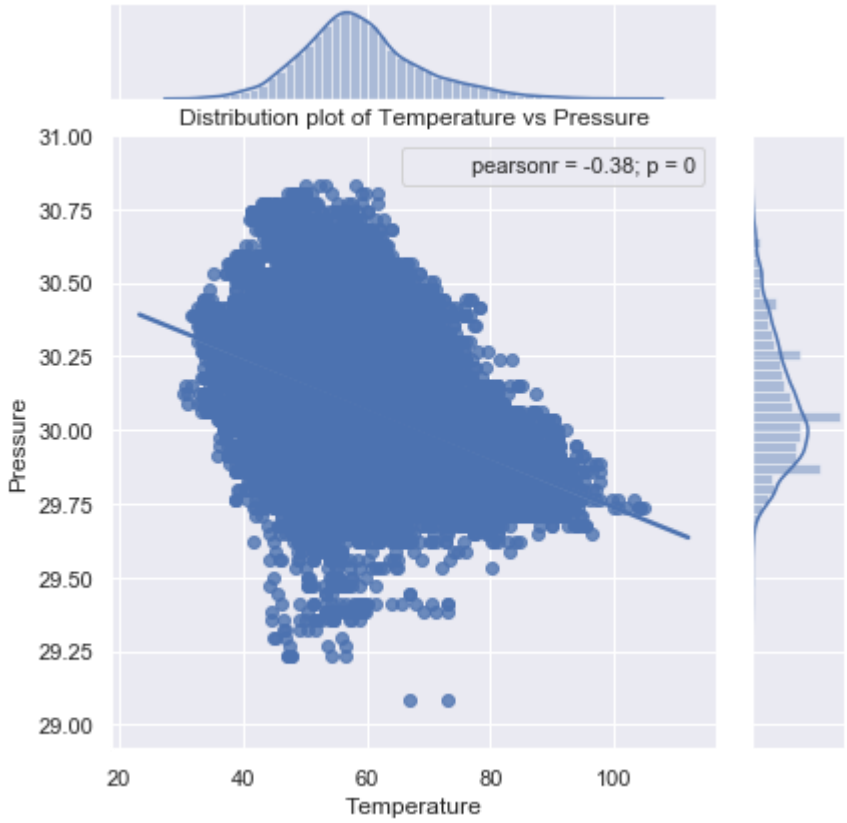
3. Constant Covariance - If a covariance is not constant in a time series then the time series exhibits randomness. Additionally, time series distribution changes without any obvious pattern. This indicates that the time series time points have changing correlation.

The San Francisco Temperature Dataset is divided into 2 sets and mean, standard deviation and co-variance is calculated for each set. The result indicated that mean and standard deviation are very similar in both sets of datasets. So the time series is Deterministic and Stationary.

Also Augmented Dickey-Fuller Test from statsmodel library indicated that the temperature is a stationary time series.

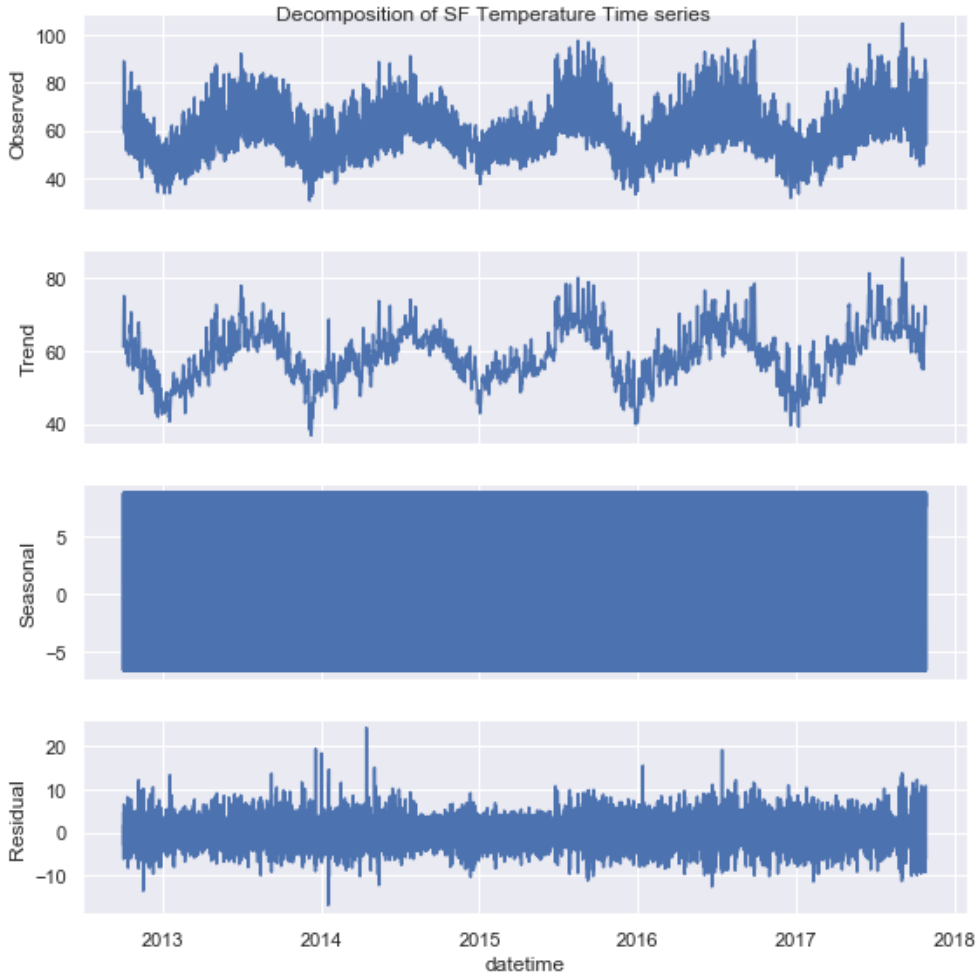
e. Time Series Histogram and Density Plot

The joint plot below shows that Temperature distribution is Normal distribution with little skewness to the right. Which could indicate outliers in higher temperatures. Pressure distribution is also normal distribution with little skewness to the left. Pearson coefficient is -0.38 which means they are inversely co-related.



f. Seasonal Decomposition

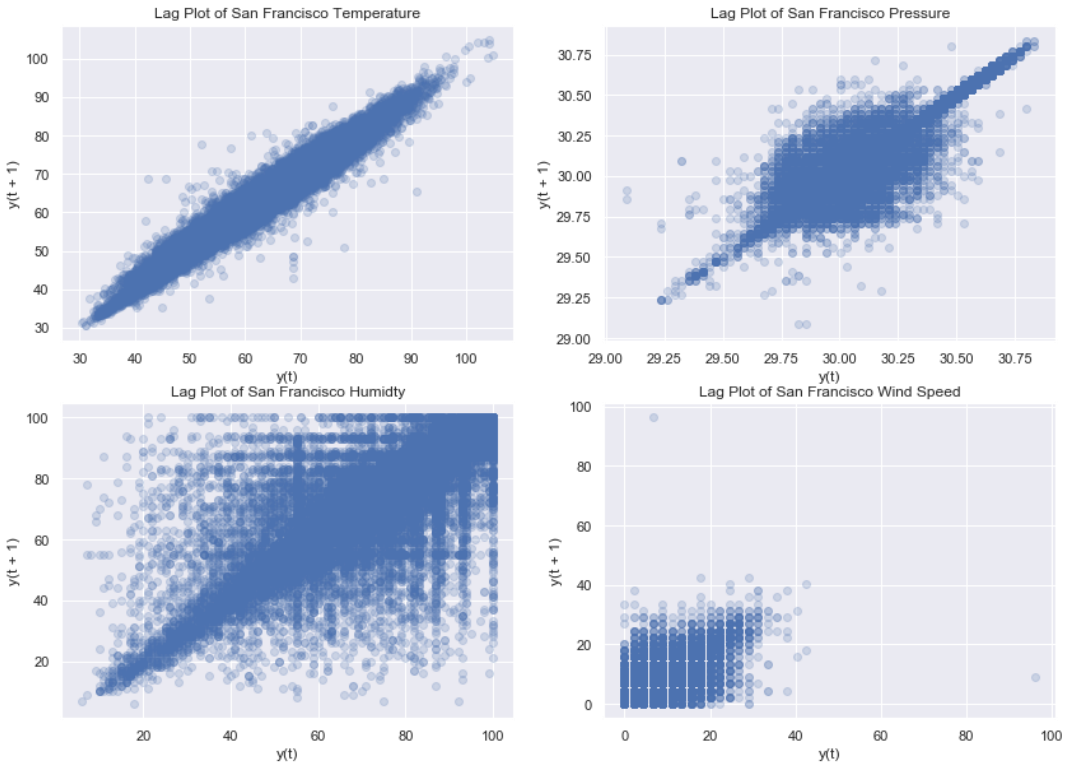
Using time-series decomposition makes it easier to quickly identify a changing mean or variation in the data. The plots below clearly show that there is yearly (long term) trend. Also there is constant seasonality. The residuals seem random with zero mean which makes it a white noise. These can be used to understand the structure of our time-series. The intuition behind time-series decomposition is important, as many forecasting methods build upon this concept of structured decomposition to produce forecasts.



g. Lag Plot

A lag plot is a scatter plot for a time series and the same data lagged. With such a plot, we can check whether there is a possible correlation between current value and the lagged value. Following observations can be made from the plot below.

1. A linear shape shows a relatively strong positive correlation between observations and their lag1 values.
2. Also it suggests that an autoregressive model is probably a better choice.
3. Outliers are easily discernible on a lag plot. The plot shows that there are several outliers.

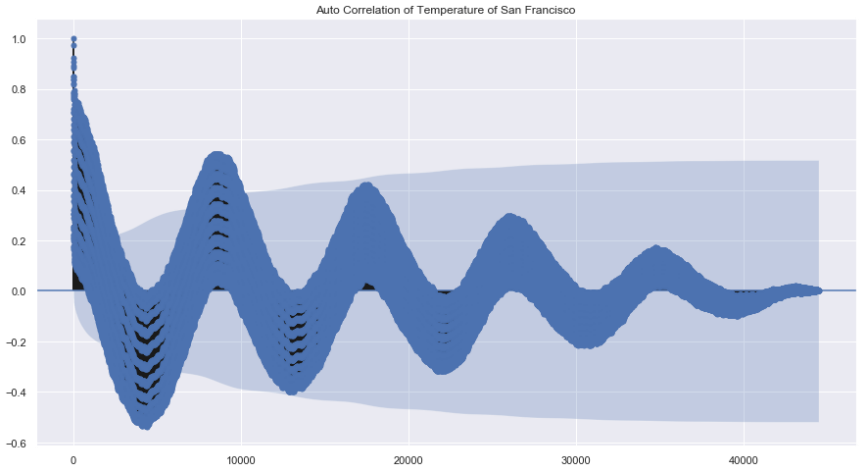


h. Auto Correlation Factor Plot

We can calculate the correlation for time series observations with observations with previous time steps, called lags. Because the correlation of the time series observations is calculated with values of the same series at previous times, this is called a serial correlation, or an autocorrelation.

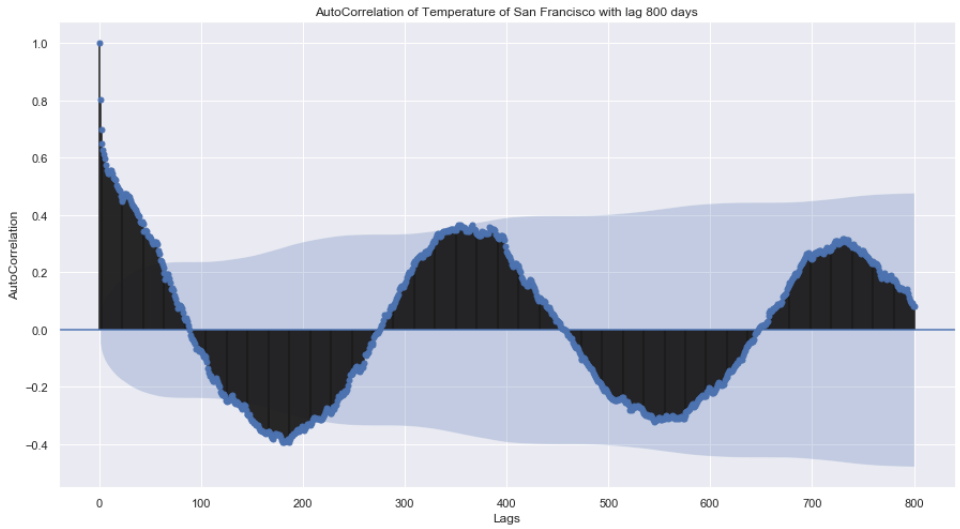
ACF Plot or Auto Correlation Factor Plot is generally used in analyzing the raw data for the purpose of fitting the Time Series Forecasting Models. ACF is used in tandem with PACF (Partial Auto Correlation Factor) to identify which Time series forecasting model to be used.

The plot below shows the lag value along the x-axis and the correlation on the y-axis between -1 and 1. Confidence intervals are drawn as a shaded cone. By default, this is set to a 95% confidence interval, suggesting that correlation values outside of this cone are very likely a correlation and not a statistical fluke. The below plot shows the with a smaller lag value. All the points except are first point are outside the confidence interval. So they are statistically significant.



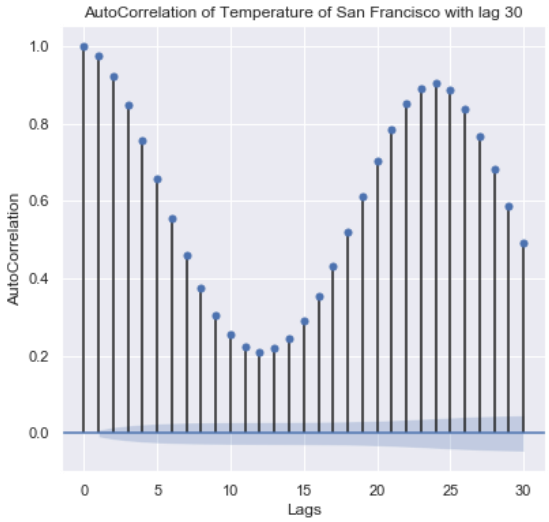
i. Auto Correlation Factor Plot

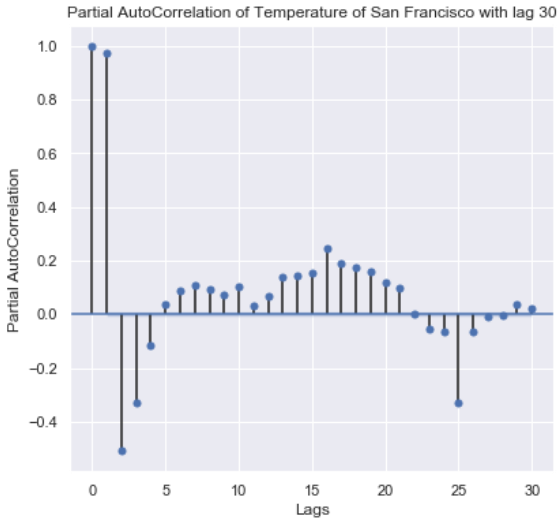
A partial autocorrelation is a summary of the relationship between observations in a time series with observations at prior time steps with the relationships of intervening observations removed. The partial autocorrelation at lag k is the correlation that results after removing the effect of any correlations due to the terms at shorter lags. In the plot below all the points lie outside the confidence interval. So all the lags are statically significant.



j. Intuition of ACF and PACF

The ACF for the AR(k) time series is strong to a lag of k and the inertia of that relationship carry on to subsequent lag values, trailing off at some point as the effect is weakened. We know that the PACF only describes the direct relationship between an observation and its lag. This would suggest that there is no correlation for lag values beyond k. In the ACF plot below, the first correlation value is strong and then it slowly becomes week in subsequent lags. In PACF plot, the correlation value become negative after 2nd value. This indicates that AR model will work on this time series with a value p = 2 for ARIMA model.





k. Multivariate Analysis

Multivariate time series has more than one time-dependent variable. Each variable depends not only on its past values but also has some dependency on other variables. This dependency is used for forecasting future values.

Johansen Cointegration Test – Co-integration is a statistical property of a collection (X_1, X_2, ..., X_k) of time series variables. First, all of the series must be integrated of order d (see Order of integration). Next, if a linear combination of this collection is integrated of order less than d, then the collection is said to be co-integrated.

In the Johansen test, we check whether lambda has a zero eigenvalue. When all the eigenvalues are zero, that would mean that the series are not co-integrated, whereas when some of the eigenvalues contain negative values, it would imply that a linear combination of the time series can be created, which would result in stationarity.

The results of the Johansen Co-integration Test show that 4 vectors of 4 weather attributes are co-integrated.

9. Deep Neural Networks

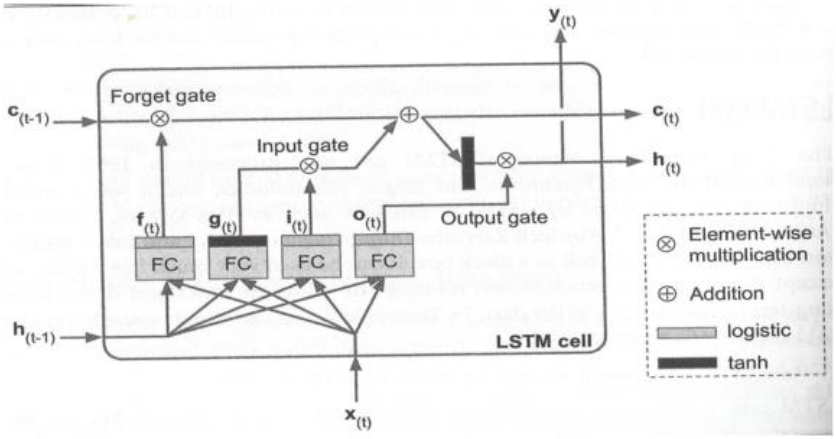
With the advent of Deep Learning algorithms, various supervised and unsupervised models are developed to analyze massive size time series data. Deep Learning is the general term for a series of multi-layer architecture neural networks. Several different Deep Learning approaches are there for time series analysis, such as Recurrent Neural Networks (RNN), The Long Short Term Memory (LSTM) neural network, Gated Recurrent Units (GRU), Stacked Auto Encoders, Multi-layer Perceptron (MLP) Regression. RNNs have shown their success in time series forecasting in recent years. However, they suffer from the problem of vanishing gradients and thus have difficulty capturing long term dependencies The LSTM and GRU have overcome this limitation and became very popular architectures for time series problems. LSTM neurons keep a context of memory within their pipeline to allow for tackling sequential and temporal problems without the issue of the vanishing gradient affecting their performance. The GRU model is simplified version of the LSTM model and performs just as well as LSTM which makes it popular architecture too for time series analysis.

10. The Long Short Term memory networks

The Long Short Term Memory networks – usually just called "LSTMs" – are a special kind of Recurrent Neural Networks, introduced by Hochreiter & Schmidhuber (1997) *[7]*. Remembering information for long periods of time is practically the default behavior of LSTM networks, not something they struggle to learn!

Below the flow diagram of LSTM cell. The key idea is that the LSTM cell can learn to recognize an important input, store it in long term state, learn to preserve it for as long as it is needed and learn to extract it whenever it is needed. First the current input vector $x_{(t)}$ and the previous short term state $h_{(t-1)}$ are fed to the four fully connected layers. Each layer serves a different purpose.

- The main layer is the one which outputs $g_{(t)}$. This layer analyzes $x_{(t)}$ and $h_{(t-1)}$ and decides what is stored in the long term state.
- The three other layers are gate controllers for all 3 gates (input, forget and output).
  - The Forget gate controls which parts of the long term state should be erased
  - The input gate controls which part of $g_{(t)}$ should be added to the long term state.
  - The output gate controls which parts of the long term state should be read and output at this time step.
- As the long term state $c_{(t-1)}$ traverses the network from left to right, it first goes through a forget gate, dropping some memories, and then it adds some new memories selected by an input gate. The result $c_{(t)}$ is sent out. After the addition operation, the long term state is copied and passed through tanh function and then the result is filtered by the output gate. This produces the short term state $h_{(t)}$which is the cell's output for this time step $y_{(t)}$



Block Diagram of LSTM Cell

The equations below summarize how to compute the cell's long term states, short term states and outputs. Here Ws are the weight matrices and bs are the bias terms.

$$\mathbf{i}_{(t)} = \sigma(\mathbf{W}_{xi}{}^{T}\mathbf{x}_{(t)} + \mathbf{W}_{hi}{}^{T}\mathbf{h}_{(t-1)} + \mathbf{b}_{i})$$
$$\mathbf{f}_{(t)} = \sigma(\mathbf{W}_{xf}{}^{T}\mathbf{x}_{(t)} + \mathbf{W}_{hf}{}^{T}\mathbf{h}_{(t-1)} + \mathbf{b}_{f})$$
$$\mathbf{o}_{(t)} = \sigma(\mathbf{W}_{xo}{}^{T}\mathbf{x}_{(t)} + \mathbf{W}_{ho}{}^{T}\mathbf{h}_{(t-1)} + \mathbf{b}_{o})$$
$$\mathbf{g}_{(t)} = (\mathbf{W}_{xg}{}^{T}\mathbf{x}_{(t)} + \mathbf{W}_{hg}{}^{T}\mathbf{h}_{(t-1)} + \mathbf{b}_{g})$$
$$\mathbf{c}_{(t)} = \mathbf{f}_{(t)} \otimes \mathbf{c}_{(t-1)} + \mathbf{i}_{(t)} \otimes \mathbf{g}_{(t)}$$
$$\mathbf{y}_{(t)} = \mathbf{h}_{(t)} = \mathbf{o}_{(t)} \otimes (\mathbf{c}_{(t)})$$

11. Deep LSTM-MLP Model

The purpose of the exercise is to build a multi-variate prediction model that learns the relationships of the temporal weather variables such as temperature, pressure, humidity, wind direction and wind direction and non-temporal variables such as latitude and longitude of locations of cities using hourly historical data of 36 different cities. To compare the results, I built a deep LSTM MLP univariate prediction model to predict the Temperature also.

A. Network Architecture

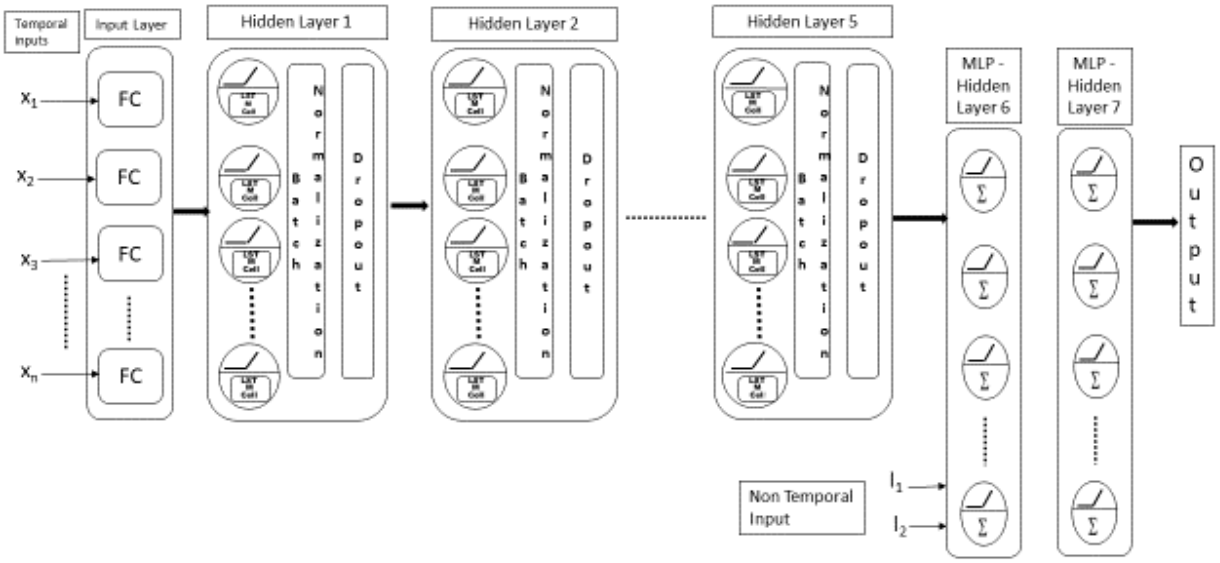
The LSTM models are very efficient models for time series forecasting problems. Deep LSTM-MLP models consists five hidden layers of a block of stacked layers. Each block has LSTM layer, a batch normalization layer and a dropout layer. The temporal inputs are applied to first LSTM layer. The output of the last dropout layer and non-temporal inputs are applied to first MLP hidden layer. Network architecture includes several DNNs features such as ReLU activation function, Glorot weight initialization, Batch Normalization, Dropouts, Adam Optimizers.

LSTM layers use Rectified Linear Unit (ReLU) activation function and Glorot weight initialization. Vanishing gradients and exploding gradients problems were empirically observed in DNNs for a long time until in *2010, Xavier Glorot and Yoshua Bengio [9]* came up with ReLU activation function and Glorot or Xavier weight initialization strategy. They said that for the signal to flow properly, the variance of the outputs of each layer should be equal to the variance of the inputs. Also, gradients should have equal variance before and after flowing through a layer in reverse direction. ReLU activation function also has an advantage for being quite fast and perform better.

Good weight initialization and activation functions solve the vanishing/exploding gradient problem to some extent but don't guarantee that they won't come back in Deep Path of training in Neural Networks. Batch Normalization, is a technique consists of adding an operation in the model in each layer, simply zero-centering and normalizing the inputs, then scaling and shifting the results before they can be given as input to the next layer. The evaluation of mean and standard deviation of the input is done over the current mini batch hence called Batch Normalization.

Dropout is another regularization technique for DNNs. At every training step, every neuron except the output neuron, has a probability p of being temporarily 'dropped out', meaning it will entirely ignored during this training step but it may be active during the next step. After each LSTM layer, a BatchNormalization and Dropout layers are added. Dropout Layer drops one percent of the inputs to the next LSTM layer.

Adam Optimizer (Adaptive moment estimation) is the most popular of all the adaptive learning rate algorithms. Algorithm calculates an exponential moving average of the gradient and the squared gradient and the parameters β_1 and β_2 control the decay rates of these moving averages. It requires less tuning of learning rate hyper-parameter and is set to the default value of 0.001 in LSTM-MLP model.



Block diagram of LSTM-MLP Deep Neural Network

B. Results Metrics

I used three metrics to compare the results. MAE, MSE and MAPE. MSE is used as the loss value for optimization during training to penalize the model for making predictions that differ greatly from the corresponding actual value. I see higher value of MAPE since MAPE grows unexpectedly large if the actual values are exceptionally small themselves. In our dataset, Wind Speed and Wind direction is close to zero for large number of samples.

C. Tools and Libraries

Different Python libraries are used for executing different Machine Learning and Deep Learning Models. LinearRegression and RandomForest from Sci-kit Learn, ARIMA model from Statsmodel, LSTM and MLP from Tensorflow Keras Functional APIs are used. The Keras functional API is the way to go for defining complex models, such as multi-output models, directed acyclic graphs, or models with shared layers. Tensorflow Estimator APIs *[8]* used for seamless execution of Deep Neural Network (DNN) in distributed multi-server environment. Estimator is a high level Tensorflow API which facilitates building state of the art, highly scalable, complex models, which can be run on local host or on a distributed multi-server environment, on CPUs, GPUs or TPUs, without changing the model. A distributed VM in Google Cloud with 8 core CPUs, 52 GB memory and 2 GPUs is used to execute DNN models.

D. Experimental Details

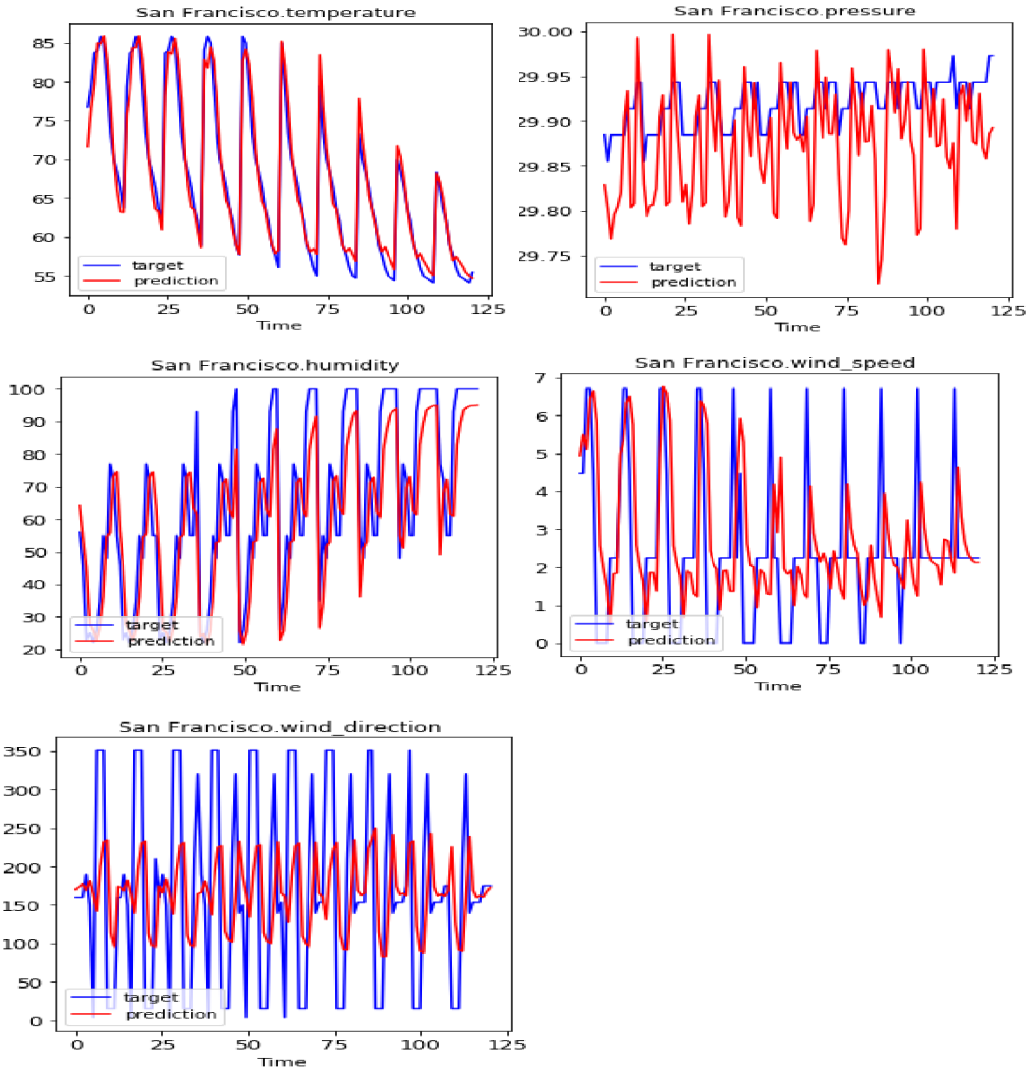
The dataset contains weather features in scientific unit and not popular in everyday life. I converted Temperature in degree Fahrenheit, Pressure in inches of Mercury and wind speed in miles per hour. These features have physical properties and related to each other such as temperature and pressure are inversely proportional to each other. Wind has both speed and direction, so can be represented by a vector. I considered these as data and not co-relating

among them is considered during the analysis. The data is normalized with zero mean and unit variance. We perform 6 step ahead prediction for all the variables for univariate analysis and multivariate analysis. Dataset is divided into training and test dataset. Test Data size is 360 which is actually 10 hours weather forecast for 36 cities. Rest of the dataset is used for the training.

12. Results

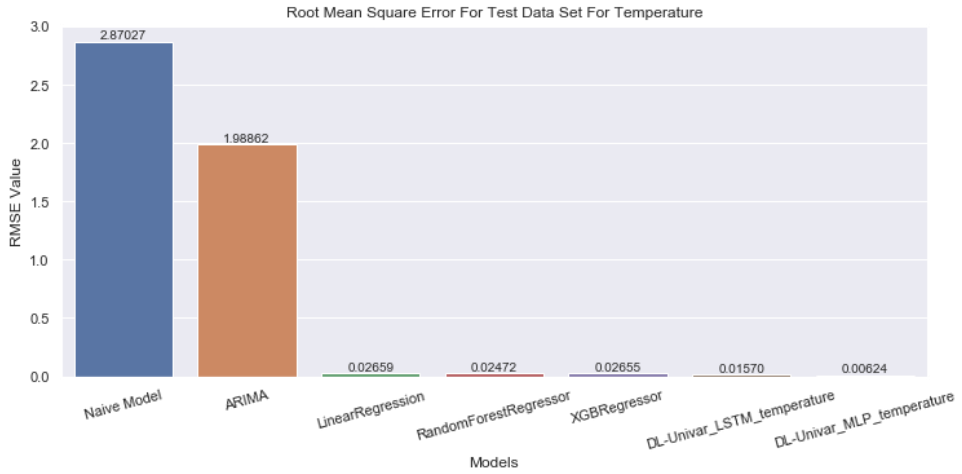
A. Qualitative Predictions

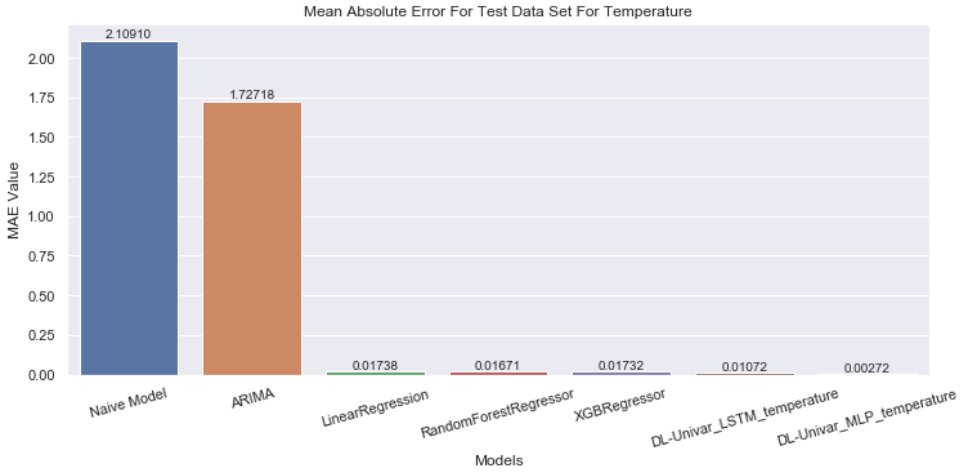
The Prediction plot for multivariate LSTM-MLP model of all five weather features for San Francisco from the weather dataset are shown below. The plot below show that Pressure and Wind Direction had higher variance and really low variance respectively and these were hard to predict. Temperature, Humidity and Wind Speed were relatively smooth and were easy to predict.



B. Aggregate Quantitative Performance

Figure below shows the values of RMSE, MAE and MAPE for uni-variable analysis of Temperature, I achieved on the hourly weather dataset. Clearly Deep Learning models Multi-Layer Perceptron (MLP) and Long Short Term Memory (LSTM) performed better Machine Learning Models for RMSE and MAE.





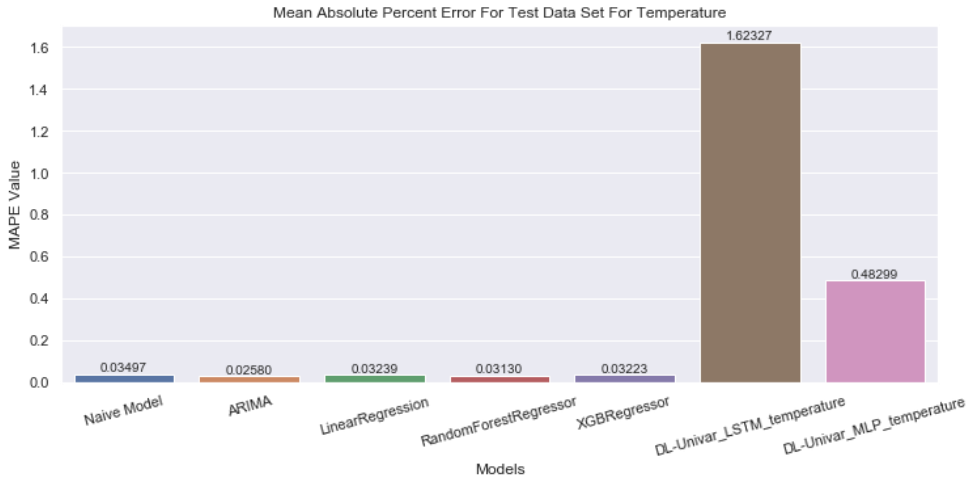
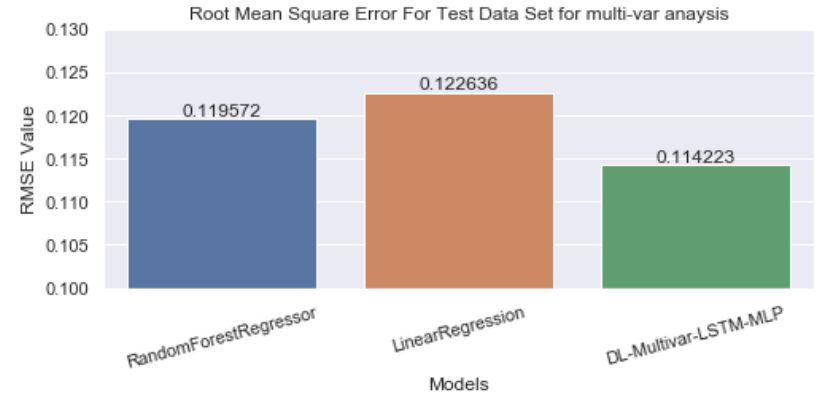
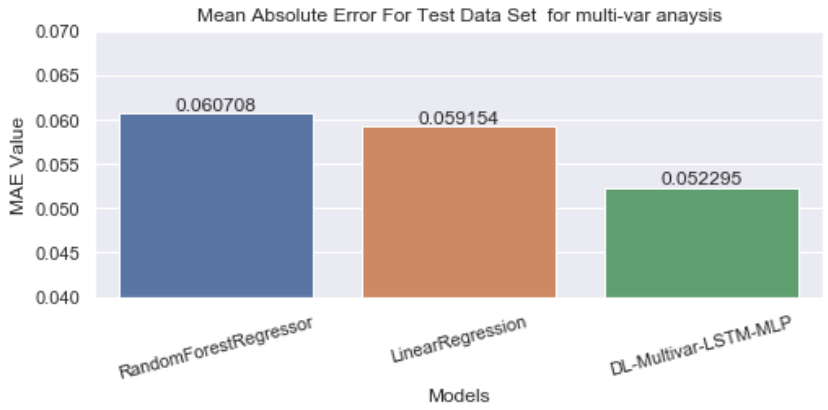
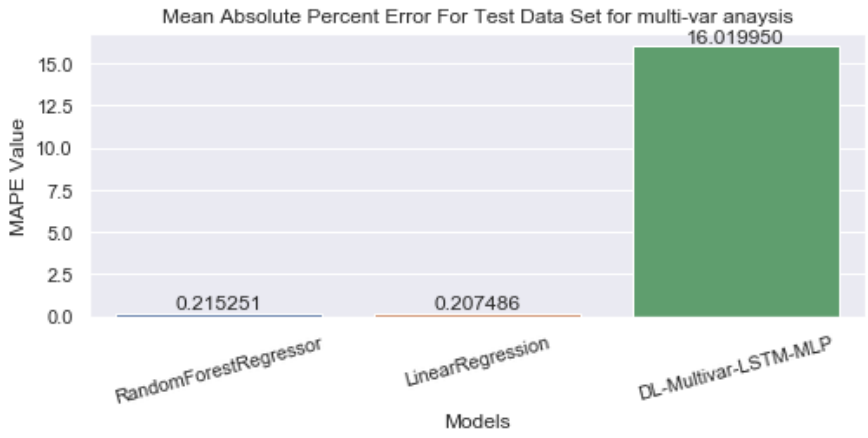


Figure below shows the values of RMSE, MAE and MAPE for multi-variable analysis of five weather features, I achieved on the hourly weather dataset. Clearly Deep Learning model LSTM-MLP performed better than Machine Learning Models for RMSE and MAE.







13. Conclusion

In this work, I presented a new model, LSTM-MLP, for multivariate time-series prediction by combining the multiple layers of LSTM cells and multiple layers of MLP layers. The idea is to use multiple LSTM layer to capture long term and short term memories of temporal weather data. Then combine that memory of non-temporal data in MLP layers to generate final predictions. I compared

the results with several Machine Learning models. Both LSTM model for univariate and LSTM-MLP model for multivariate analysis performed better than traditional machine learning model. As I increase the number of steps ahead I want to make the predictions, the complexity of the model and time require to train the model increases. Deep Learning models with distributed networking can perform better and make analysis faster.

14. References:

1. Link to data set: https://www.kaggle.com/selfishgene/historical-hourly-weather-data
2. Liu, J.N., Hu, Y., You, J.J., Chan, P.W.: Deep neural network based feature representation for weather forecasting. http://worldcomp-proceedings.com/proc/p2014/ICA3405.pdf
3. Aditya Grover∗ IIT Delhi, Ashish Kapoor Microsoft Research, Eric Horvitz Microsoft Research: A Deep Hybrid Model for Weather Forecasting:
4. http://erichorvitz.com/weather_hybrid_representation.pdf
5. John Gamboa: Deep Learning for Time-Series Analysis. https://arxiv.org/abs/1701.01887
6. Dmitry Vengertsev: Deep Learning Architecture for Univariate Time Series Forecasting: http://cs229.stanford.edu/proj2014/Dmitry%20Vengertsev,Deep%20Leraning%20Architecture%20for%20Univariate%20Time%20Series%20Forecasting.pdf
7. Understanding LSTM Networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/
8. Tensorflow Estimators. https://arxiv.org/abs/1708.02637
9. Understandingthedifficultyoftrainingdeepfeedforwardneuralnetworks: https://homl.info/47
10. Time Series Analysis in Python with statsmodels: https://www.researchgate.net/publication/265673479_Time_Series_Analysis_in_Python_with_statsmodels
11. Hourly Time Series Forecasting using XGBoost: https://www.kaggle.com/robikscube/tutorial-time-series-forecasting-with-xgboost
12. Predicting the direction of stock market prices using random forest : https://arxiv.org/pdf/1605.00003.pdf
13. Machine Learning for Financial Market Prediction — Time Series Prediction With Sklearn and Keras : https://alphaarchitect.com/2018/06/05/machine-learning-financial-market-prediction-time-series-prediction-sklearn-keras/