



# WEATHER FORECASTING APPLICATION

---

## BUILDING DS PRODUCT – STAGE 4(FINAL)

### Course:

Data Science Product Development  
(50742)

### Group members:

Arsalan Shaikh - 23388  
Ramsha Arif - 23395  
Sarosh Aamir - 23390

## Table of Contents

Business Objective .....	3
Underlying Data Science Problem .....	3
Tools & techniques .....	4
Data set.....	5
Models / techniques .....	6
Regressor Models .....	6
1.    Random Forest:.....	6
2.    Decision Tree: .....	6
Time-Series Models .....	6
3.    Prophet: Automatic Forecasting Procedure .....	6
4.    Long Short-Term Memory.....	7
Quantization on LSTM.....	8
i.    Dynamic Range Quantization: .....	8
ii.   Quantization Aware Training:.....	8
Ensembling .....	10
Dockerization .....	11
Fast API Implementation.....	11
AirFlow Implementation .....	12
Final Model Prediction Screenshot .....	12
References.....	13

---

# BUSINESS OBJECTIVE

Weather forecasting is the application of scientific techniques and technology to predict the conditions of the atmosphere at a certain location and time. Weather Forecasting in old time is carried out by hand, using changes in barometric pressure, current weather conditions, and sky condition or cloud cover, weather forecasting now relies on computer-based models that take many atmospheric factors into account. accounting now relies on computer-based models that take many atmospheric factors into account.

Our main goal is to analyze how the different machine learning techniques will perform in the forecasting of weather combining time series with the neural network.

This project is divided into two parts, namely a research component and an application component, aiming to provide users with weather **predictions** using different machine learning models along with providing a good user experience with **interactive interface**.

---

# UNDERLYING DATA SCIENCE PROBLEM

In this case, a software system can learn from data for improved analysis. Compared to traditional demand forecasting methods, machine learning helps to:

- Accelerates data processing speed
- Provides a more accurate forecast
- Automates forecast updates based on the recent data
- Analyzes more data
- Identifies hidden patterns in data
- Creates a robust system
- Increases adaptability to changes

---

# TOOLS & TECHNIQUES

Following are the details that have been achieved successfully and has been implemented practically:












- **Dataset-** hosted on S3 bucket (AWS).
- **Python-** is being used as a programming language.
- **Boto3-** has been used to create, configure, and manage AWS services into colab.
- **Fast API-** is used for building APIs.
- **Docker Container** – is be used to manage application separately.
- **Other Libraries-** Pandas, Scikit-Learn, TensorFlow, Keras etc.
- **Regression Models-** Random Forest and Decision Tree
- **Time-Series Based-** Fb Prophet and LSTM
- **Quantization-** Dynamic Range and Quantization Aware Training
- **Pipeline-** Dagster, Airflow

# DATA SET

- The dataset that has been picked from World Weather Online [website](#).
- It provides us historical data as well as gives us information with respect to location that we choose. Here, “Karachi, Pakistan” data has been collected.
- The source data contain various features that can help to find optimum prediction for the data. Moreover, time-series data is used to obtain accuracy and apply some advance Machine Learning and Deep Learning algorithms.
- Through their API, last 11 years historical data has been fetched.
- After which we have converted the JSON response into .csv format, segregating it into each year's file.
- All of these files have been **stored in Amazon S3 Bucket**. By using boto3, the csv data was called from S3 and converted it into our dataset.

Objects (11)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	 <a href="#">2010.csv</a>	csv	March 10, 2021, 16:11:03 (UTC+05:00)	449.6 KB	Standard
<input type="checkbox"/>	 <a href="#">2011.csv</a>	csv	March 10, 2021, 16:11:48 (UTC+05:00)	899.7 KB	Standard
<input type="checkbox"/>	 <a href="#">2012.csv</a>	csv	March 10, 2021, 16:12:37 (UTC+05:00)	1.3 MB	Standard
<input type="checkbox"/>	 <a href="#">2013.csv</a>	csv	March 10, 2021, 16:13:28 (UTC+05:00)	1.8 MB	Standard
<input type="checkbox"/>	 <a href="#">2014.csv</a>	csv	March 10, 2021, 16:14:20 (UTC+05:00)	2.2 MB	Standard
<input type="checkbox"/>	 <a href="#">2015.csv</a>	csv	March 10, 2021, 16:15:23 (UTC+05:00)	2.6 MB	Standard
<input type="checkbox"/>	 <a href="#">2016.csv</a>	csv	March 10, 2021, 16:16:21 (UTC+05:00)	3.1 MB	Standard
<input type="checkbox"/>	 <a href="#">2017.csv</a>	csv	March 10, 2021, 16:17:16 (UTC+05:00)	3.5 MB	Standard
<input type="checkbox"/>	 <a href="#">2018.csv</a>	csv	March 10, 2021, 16:02:05 (UTC+05:00)	451.5 KB	Standard
<input type="checkbox"/>	 <a href="#">2019.csv</a>	csv	March 10, 2021, 16:02:51 (UTC+05:00)	903.6 KB	Standard
<input type="checkbox"/>	 <a href="#">2020.csv</a>	csv	March 10, 2021, 12:56:05 (UTC+05:00)	436.1 KB	Standard

- Dataset consists of following files:
  - 🕒 **Elements:** date, time, temperature, feels like, humidity, dew point, wind speed, wind direction etc.
  - 🕒 **Format:** .csv format.
  - 🕒 **Date Range:** 2010-2020

---

# MODELS / TECHNIQUES

## Regressor Models

### 1. Random Forest:

A **random forest** is a meta estimator that fits a number of classifying **decision** trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

#### Analysis:

The difference between the actual and predicted values is the very minor, so this model could be used for the out of sample predictions.

### 2. Decision Tree:

Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed.

#### Analysis:

This model mapped the predicted values accurately and hence the difference between actual and predicted values was almost equal to zero. The model seems to be perfect to predict the out of sample values, but this may have an overtraining issue as this is perfectly mapping the training data.

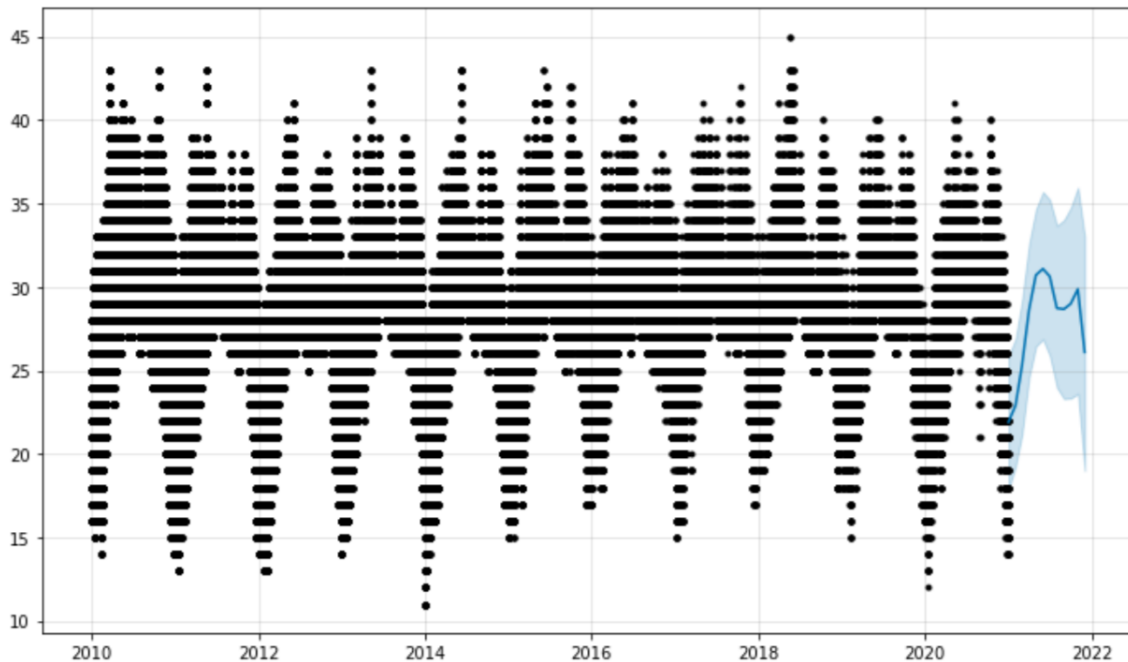
## Time-Series Models

### 3. Prophet: Automatic Forecasting Procedure

Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data. Prophet is robust to missing data and shifts in the trend, and typically handles outliers well.

#### Analysis:

There is an uncertainty in prediction, it is not completely accurate. Further analysis is needed to optimize the model.



2-years prediction using Prophet

## 4. Long Short-Term Memory

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. An LSTM is generally used favorably for time-series data (such as our precipitation datasets) as it is able to ‘remember’ long-term dependencies/information from a feature called cell state. This cell state runs across the networks and passes down (or up) information such that earlier information from a sequence, for instance, could influence prediction of later input. This passing of information is crucial for time-series weather data where each input is dependent on other input from varying time points. The information that is stored, removed, modified, and passed around is controlled by gates.

### Data Preprocessing:

Here we are picking 350616 records for training. Observation is recorded every day, that means 24 times per day. That means the no. of days we have,

`350616 / 24 equals to 14609 days`

For training data, we are using 75% of days,

`14609/0.75 which is equals to 10957 days`

For testing the remaining 25% of days,

`14609-10957 = 3652`

We will resample one point after every 3 hours since no drastic change is expected within 3 hours. We do this via the **sampling\_rate** argument in **timeseries\_dataset\_from\_array** utility.

The **timeseries\_dataset\_from\_array** function takes in a sequence of data-points gathered at equal intervals, along with time series parameters such as length of the sequences/windows, spacing between two sequence/windows, etc., to produce batches of sub-timeseries inputs and targets sampled from the main timeseries.

#### Analysis:

So far, multiple iterations have been performed on single-cell LSTM. To see the variation on different iterations, go to this link:

[https://drive.google.com/file/d/1khU2QhoeZIKAHhMaA4oD\\_S-R8qzMfjMS/view?usp=sharing](https://drive.google.com/file/d/1khU2QhoeZIKAHhMaA4oD_S-R8qzMfjMS/view?usp=sharing)

The training loss shows is minimal and Mean-Squared Error and R2 calculated on testing dataset depicts the accuracy of a model in high range value.

Simple LSTM Iterations							
Iteration	Epochs	Batch size	Learning Rate	Model	Test Loss	MSE	R2
1	5	32	0.001	Single Cell LSTM	0.0194	11.59	0.32
2	15	32	0.001	Single Cell LSTM	0.0112	8.39	0.5
3	10	128	0.002	Single Cell LSTM	0.0224	5.67	0.66

## Quantization on LSTM

Quantization is a process of approximating a neural network that uses floating-point numbers by a neural network of low bit width numbers. This technique helps reduce both the memory requirement and computational cost of using neural networks at the cost of modest decrease in accuracy.

Furthermore, TensorFlow provides an open-source deep learning framework for on-device inference called TF Lite. TF Lite is the lightweight version of TensorFlow, specifically designed for the mobile platform and embedded devices, that allows for running a large model with low latency and small binary size. Furthermore, TensorFlow Lite converter also provides options to perform post-training quantization during model conversion.

There are two different methods of quantization which are implemented to achieve higher accuracy:

### i. Dynamic Range Quantization:

Here we will be performing Dynamic Range Quantization which is a Post-training quantization. This technique can be applied while converting the trained TF model to Tf Lite version, using the optimization options provided by the TensorFlow Lite converter. In this method we statically quantize the weights from floating point to 8-bits of precision and dynamically quantizes the activations at inference. This means that the activations are always stored in float 32, however, they are converted to 8-bit integers while processing and back to floating point after the processing is done.

### ii. Quantization Aware Training:

QAT models quantization during training and typically provides higher accuracies as compared to post-training quantization. QAT is achieved by adding fake quantization nodes (where float values are approximated as 8-bit integers) at both training and inference.



There are two ways for performing QAT: Either quantizing whole model or some layers. Here we have quantized whole model.

### Analysis Post-Quantization:

After implementing Quantization, the **weight** of a model has reduced evidently in both quantization methods while the **accuracy** has dropped drastically especially in Dynamic Range Quantization the output value was replaced to the average value points that makes the model erroneous.

However, the second method has shown far better results ~40% accuracy.

The **timing** of training a model has decreased which makes it work faster using minimum resources.

Quantization Iterations								
Iteration	Epochs	Batch size	Learning Rate	Method	Test Loss	RMSE   R2	Accuracy	Model Size in MB
1	10	32	0.0001	Post Training Dynamic Range Quantization	0.175	0.42   -11.21	19.5153	Post-Training: 20.587     Tf.lite: 6.853     Dynamic Quantized: 1.72
2	10	32	0.0001	Quantization Aware Training	0.175	0.42   -11.21	43.112	Post-Training: 20.587     Tf.lite: 6.853     QAT Quantized: 1.721

Link to the Colab file:

[https://colab.research.google.com/drive/1J\\_1oBUODaHoh1T4ozXqqncFCDA6X0Znt?usp=sharing](https://colab.research.google.com/drive/1J_1oBUODaHoh1T4ozXqqncFCDA6X0Znt?usp=sharing)

---

# ENSEMBLING

In weather forecast, we ensemble set of forecasts that present the range of future weather possibilities. Multiple simulations are run, each with a slight variation in model of its initial conditions and with slightly perturbed weather models. These variations represent the inevitable uncertainty in the initial conditions and approximations in the models. They produce a range of possible weather conditions.

The main purpose of this ensemble is to try to assimilate the initial uncertainty (initial error) and the forecast uncertainty (forecast error) by applying initial perturbation method on training models.

Here, we are using predictions retrieved from four different models:

- 1) Random forest
- 2) Decision Tree
- 3) FB Prophet
- 4) LSTM

We have taken last 10 days output from each model.

For eg. T1 – T10 for M1, M2, M3, M4

Then multiplied it with some weight to add some uncertainty in predicted temperature.

For eg. T1w1, T2w1,....., T10w1

Since, the output given by trees are not in a sequential form therefore the ensembling of trees has done separately.

While, Fb Prophet and LSTM has grouped together.

Ensembling has not been included in fastAPI because this work has been done separately just to cater the requirements and also to determine the change in forecast if the uncertainty occurs.

Link to Ensembling file:

<https://colab.research.google.com/drive/1wCjJL6zS8yOWJtPt-oujAl0IU0N9B4OC?usp=sharing>

---

# DOCKERIZATION

Docker is an open platform for developing, shipping, and running applications. Dockerization enables you to separate your applications from your infrastructure so you can deliver software quickly. As per the requirement, a docker image has been created in which all the necessary libraries are mentioned to be installed under the Dockerfile. We have used dockerization to dockerized our models and FastAPI.

---

# FAST API IMPLEMENTATION

FastAPI is a modern, fast (high-performance), web framework for building APIs with Python. It is used in this project as the model is exported and wrapped around Fast API. Through which, using Fast API's endpoint will be able to receive date for which prediction is required and will return output to the user.

In our project, it is used for following purposes:

- 1) Used to display the weather prediction
- 2) Load model into the memory after docker is started
- 3) Scheduled to download latest model from AWS.
- 4) User can route to “/lstm” to get future 5 days prediction.
- 5) Google Charts to display the line graph.

# AIRFLOW IMPLEMENTATION

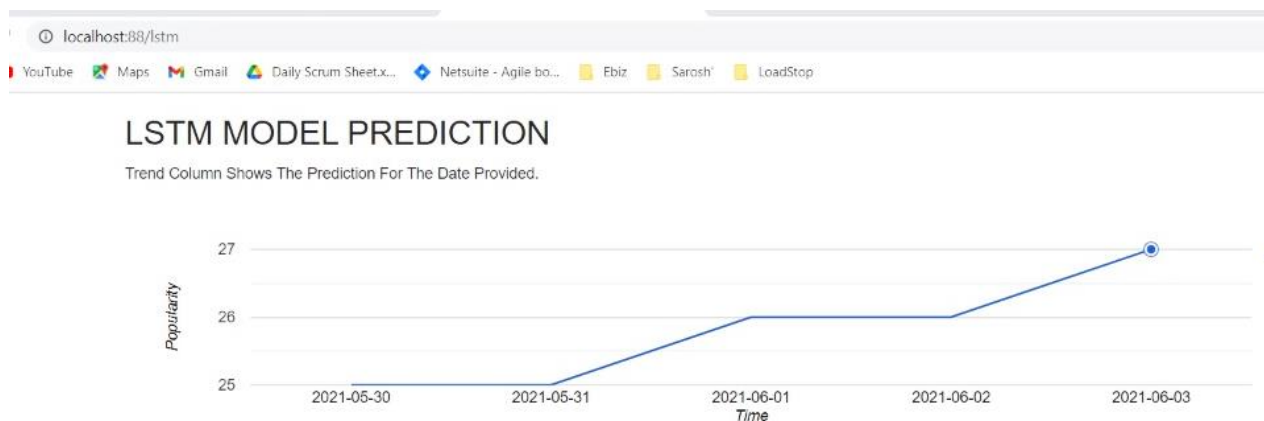
Airflow is a platform to programmatically author, schedule and monitor workflows. It is used to author workflows as Directed Acyclic Graphs (DAGs) of tasks. The Airflow scheduler executes your tasks on an array of workers while following the specified dependencies. Rich command line utilities make performing complex surgeries on DAGs a snap. The rich user interface makes it easy to visualize pipelines running in production, monitor progress, and troubleshoot issues when needed.

When workflows are defined as code, they become more maintainable, versionable, testable, and collaborative.

The steps we have taken for Airflow implementation are:

- 1) It is used to create a DAG.
- 2) There are three main components of pipeline segregated into three stages:
  - a. **First Stage (Data Preprocessing):** Obtain the data from S3, Preprocess.
  - b. **Second Stage(Training):** Dependent on First Stage. Receives the data, train the LSTM Model.
  - c. **Third Stage (Push The Model):** Dependent on Second Stage. Receives the Trained Model and pushes that into the Cloud (AWS).
- 3) It is scheduled to run every day.

## Final Model Prediction Screenshot



---

# REFERENCES

<https://colab.research.google.com/gist/NobuoTsukamoto/b42128104531a7612e5c85e246cb2dac/qat-in-keras.ipynb#scrollTo=nliAfyqCTKN8>

[https://www.tensorflow.org/lite/performance/post\\_training\\_integer\\_quant](https://www.tensorflow.org/lite/performance/post_training_integer_quant)

<https://machinelearningmastery.com/multivariate-time-series-forecasting-lstms-keras/>

<https://www.rs-online.com/designspark/predicting-weather-using-lstm>

<https://machinelearningmastery.com/save-load-keras-deep-learning-models/>

<https://ramdesh.medium.com/convenience-functions-in-python-for-saving-keras-models-directly-to-s3-b743f4dcea1f>