**Time series Modeling**

ML-based time-series forecasting to predict future methane concentrations at the regional level.

In order to predict methane concentrations for the next three to five years on a district level we employed a time-series forecasting approach using Prophet library in Python.

There are many models present for the predictive analysis of time series like Machine learning ARIMA (Auto-Regressive Integrated Moving Average model), Auto-Regressive model, Exponential Smoothing, LSTM (Long Short-Term Memory), etc. These models require the data to be fed and with certain tweaking and fine-tuning they help us to make predictions.

**Prophet** was launched by Facebook as an API for carrying out the forecasting related things for time series data. The library is so powerful that it has the capability of handling stationarity within the data and also seasonality related components. By stationarity, we mean that there should be constant mean, variance, and covariance in the data if we divide the data into segments with respect to time and seasonality means the same type of trend the data is following if segregated based on time intervals.

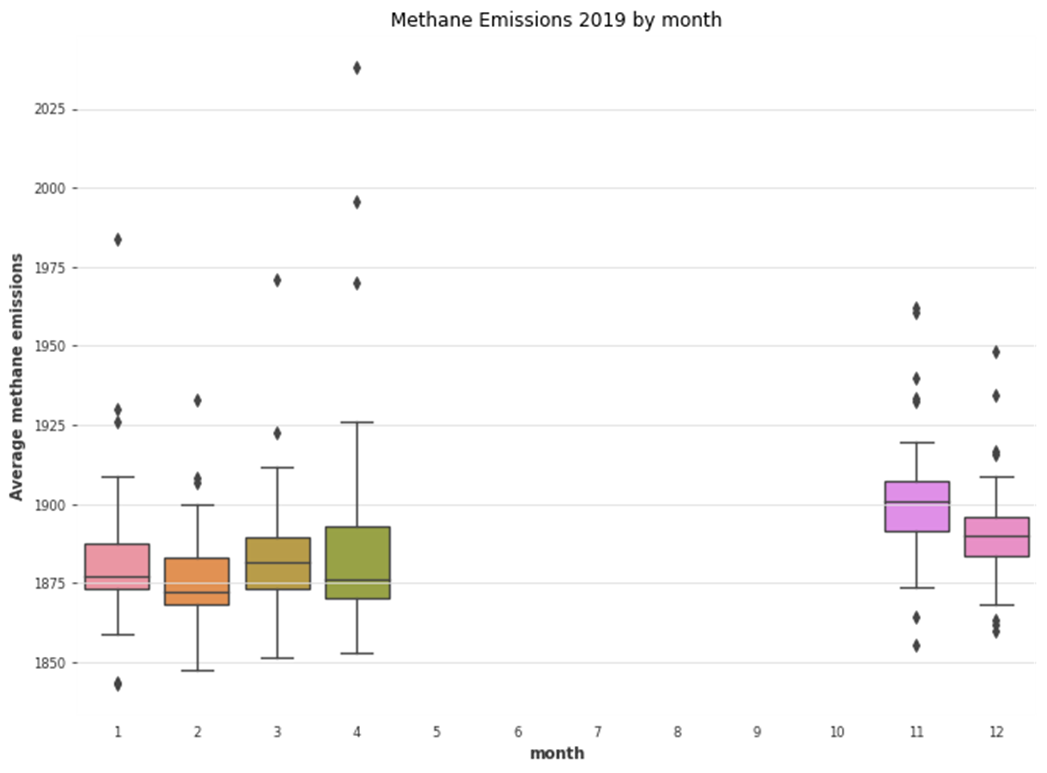
This library offers n number of parameters to improve model performance and tune our model with higher efficiency for e.g., specifying holidays, daily seasonality, Fourier transformations, etc.

This study performed a comparative approach to forecast methane emissions from the annual data from 2019-2021 which was gathered from CPE team and used famous time-series model PROPHET model. We used NAÏVE forecasting model as baseline model to compare performance of the Prophet model. The models used that were evaluated using performance metrics that gave MAE 20.5. Using this best model, average methane emissions were also forecasted from 2022 to 2024. The idea of this post is to use a univariate time-series dataset and produce a best-fit model that allows us to confidently predict future production.

Below are some snapshots of the overall Model building process starting from data exploration to Model validation. More Detailed instructions about end-end TS analysis can be found in the Jupyter notebooks shared.

**Exploratory data analysis**

A sample box plot showing distribution of methane concentration for the year 2019 by month.

****

**Time series Assumptions before training**

**Auto correlation**

These are important plots for time series. They graphically summarize the strength of the relationships of observations in time series.

In Autocorrelation, we calculate the correlation for time-series 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, hence are called a serial correlation or an autocorrelation. The processed data was checked for autocorrelation plot before model training.

**Stationarity of Data:**

Stationarity is an essential concept in time series analysis. If our data is Stationary, it means that the summary statistics of our data (or rather the process generating it) are consistent and do not change over time. It is important to check the stationarity because many useful analytical tools and statistical models rely on it. I have DF methods below:

**Dickey-Fuller test:**

One of the best methods to check for stationarity in our data is statistical tests. I have used The Augmented Dickey-Fuller test which is called a unit root test. The null hypothesis of the test is that the time series can be represented by a unit root concluding our data as not stationary.

p-value > 0.05: Fail to reject the null hypothesis (H0), the data has a unit root and is non-stationary.

p-value <= 0.05: Reject the null hypothesis (H0), the data does not have a unit root and is stationary.

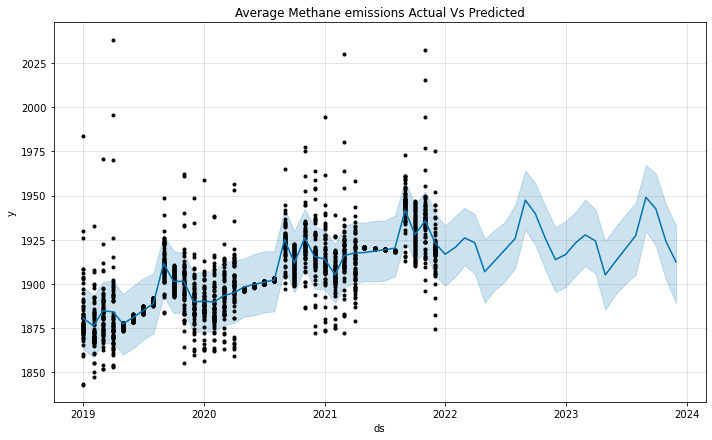
**Missing data Treatment:**

A Python Function was written to impute data for September by 2.2% than annual minimum based on literature so the model is forced to predict an increase during monsoons. (https://agupubs.onlinelibrary.wiley.com/doi/full/10.1002/2016JG003740)

**Missing data Treatment**

Linear interpolation and KNN imputation

**Forecast – Prophet Actual and Predicted with forecast for two years**



# Robustness of Model

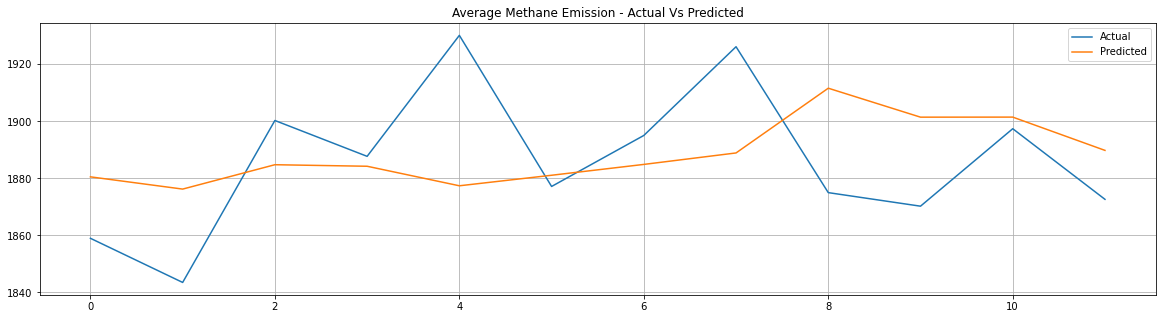
# Next, we will check the model robustness using the best metrics for measuring accuracy of this model. Utilizing a combination of R-Squared, Mean Squared Error and Mean Absolute Error will help us to gauge the quality of our model. We will Python’s Scikit-Learn library to quickly calculate these metrics.

# R-squared (R2) is a statistical measure that represents the proportion of the variance for a dependent variable that’s explained by an independent variable or variables in a regression model.

# Mean Absolute Error (MAE) measures the average magnitude of the errors in a set of predictions, without considering their direction. It’s the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

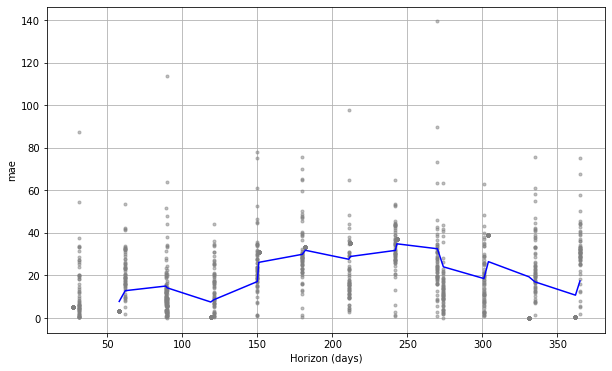
# Visualize Actual vs Predicted Values

# Finally, we create a plot to compare actual vs. predicted values to give a clear understanding of how our model visually looks against the existing Methane emissions dataset.



**Cross validation**

Prophet includes functionality for time series cross validation to measure forecast error using historical data. This is done by selecting cutoff points in the history, and for each of them fitting the model using data only up to that cutoff point. We can then compare the forecasted values to the actual values. This figure illustrates a historical forecast on the dataset, where the model was fit to a initial history of 3 years, and a forecast was made on a one year horizon.



**Hyperparameter tuning**

Cross-validation can be used for tuning hyperparameters of the model, such as changepoint\_prior\_scale and seasonality\_prior\_scale. Here parameters are evaluated on MAE averaged over a 30-day horizon, but different performance metrics may be a appropriate for different problems.

# 

More detailed comments and instructions are added in Jupyter notebook.