Project Report:
Prediction of Particulate Matter (PM2.5) in Lucknow, India

Shaunak Phatak

Problem Context

Air pollution is a serious global health hazard accounting for 8.8 million deaths annually[1]. It is a risk factor for various ailments such as heart disease, lung cancer and respiratory ailments[2]. Among the worst hit, India ranks in the top 10 countries and also has the unfortunate distinction of 21 out of 30 cities in the world with the poorest air quality in terms of particulate matter (PM2.5).

Particulate matter (PM) is among the worst types of air pollutants. The term represents a microscopic mixture of solid and liquid matter suspended in air. These particles are generated from various sources some of which include direct sources such as construction sights, fires or from complex reactions of chemicals like sulphur dioxide, nitrogen dioxide emitted by power plants and automobiles[3]. The types of PM are designated by their diameters such as PM10 and PM2.5. The smaller size of these particles enables them to penetrate directly to the lungs and bloodstream leading to various respiratory illnesses. Out of the 1.67 million deaths in India in 2019, the majority (0.98 million) were attributed to particulate pollution[4] and economically led to a loss of $36.8 billion or 1.36% of India's GDP.

Problem Statement

The goal of this project is to use publicly available data from the Central Pollution Control Board, India (CPCB) as well as weather data from the National Oceanic and Atmospheric Administration (NOAA) to create a model to forecast PM2.5. I chose Lucknow as the target city for it being among the top 30 polluted cities in the world and availability of data. This work can be further extended to create a real time forecasting tool to warn the public about potential health hazards and help take some necessary precautions such as using appropriate protective face coverings during certain days as an example.

The dataset used was as a multivariate time series available on an hourly frequency. The problem was solved by transforming the time series to a supervised learning format so as to use standard machine learning algorithms. In further sections, I will cover in detail the steps that were taken as part of a structured data science pipeline.

Data Wrangling and Exploratory Data Analysis

The dataset collected from CPCB included hourly measurements for the parameters: PM2.5, sulphur dioxide, ozone, carbon monoxide, nitrogen dioxide, nitrogen oxide, wind speed, wind direction, solar radiation and atmospheric pressure. Quarterly files were available to download for each of the above parameters. These files were merged with hourly atmospheric and dew temperature measurements from the NOAA to create a complete dataset for the project after cleaning steps such as removing unnecessary rows like IDs, standardizing column names as well as converting data from a string to numeric format. The weather data from NOAA had many duplicate hourly values which were grouped to a single mean value. Other methods tested included using the first or last value which did not show any significant changes in the distribution of temperatures across these three approaches.

The findings as part of exploratory data analysis are discussed below:

For missing data, rows with missing values for PM 2.5 were dropped as that was the variable being predicted. For other feature variables, missing values were imputed by the respective column means. Also, checking the summary statistics showed corrupted data marked as 999 as well as negative entries for some parameters. Both these types of rows which could have occurred due to instrumentation errors were replaced with a missing (NaN) token and imputed later with a column mean.

For outlier treatment, multiple strategies such as setting positive and negative bounds to 3 standard deviations, capping data to a percentile value were tested. In both these cases, the data distribution changed significantly after applying these procedures. Hence, it was decided not to make any significant changes to treat outliers due to lack of enough domain knowledge to decide what value could be considered an outlier.
Looking at the time series plot for PM2.5 below, the stray value of ~2400 as seen in the figure was removed while retaining all other entries.

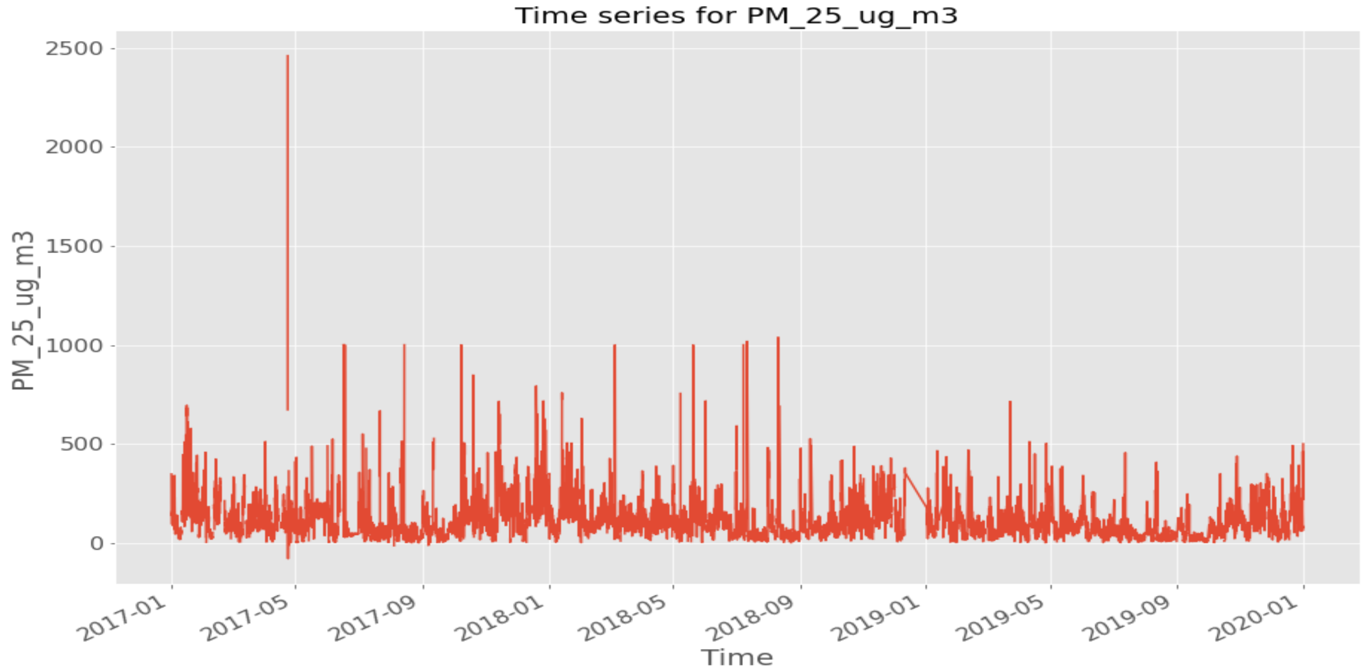

Figure 1: Time Series for particulate matter (PM2.5)

Similarly, in the case of sulphur dioxide, values <400 ug/m3 were retained while removing some of the spikes reaching 1000 ug/m3 as seen in the figure below.

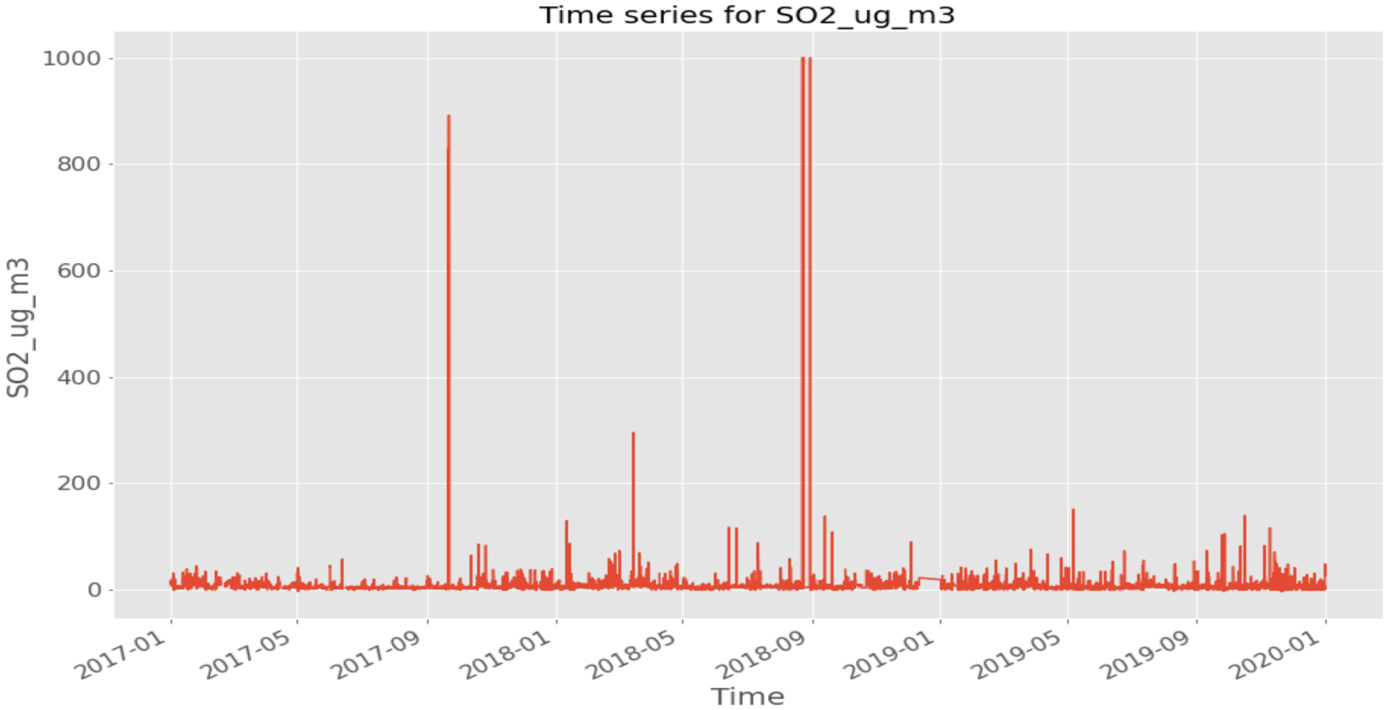

Figure 2: Time Series for sulphur dioxide

Time based features such as hour, day of week and month were extracted from the date time index. The PM2.5 values were then plotted against these parameters to check for seasonal trends.
As seen in the bar plot for average monthly PM2.5 concentration, winter months showed the highest values in the period October-January, followed by a drop between February and June during the summer months. The values were the lowest during the monsoon months of July to September.

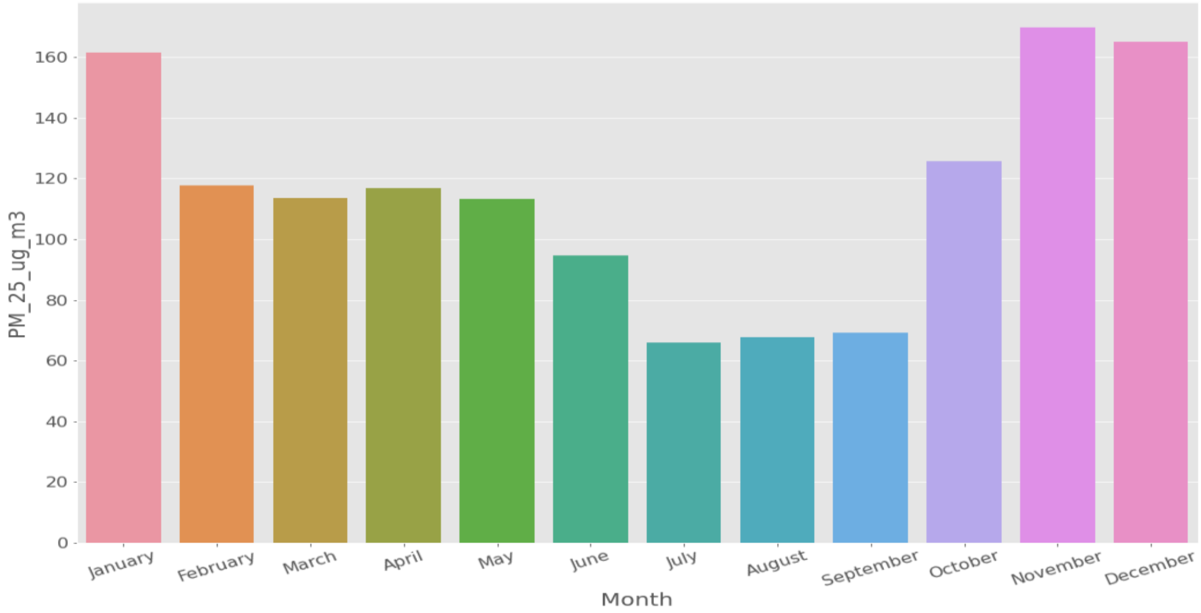

Figure 3: Bar Plot for average monthly PM2.5 Values

Similarly, as seen for hour of day, concentration values dropped slightly from midnight to 7 am, followed by a slight increase till 9 am. Post 9 am, a consistent decline can be observed till 4 pm followed by an increasing trend during the evening hours up to midnight.

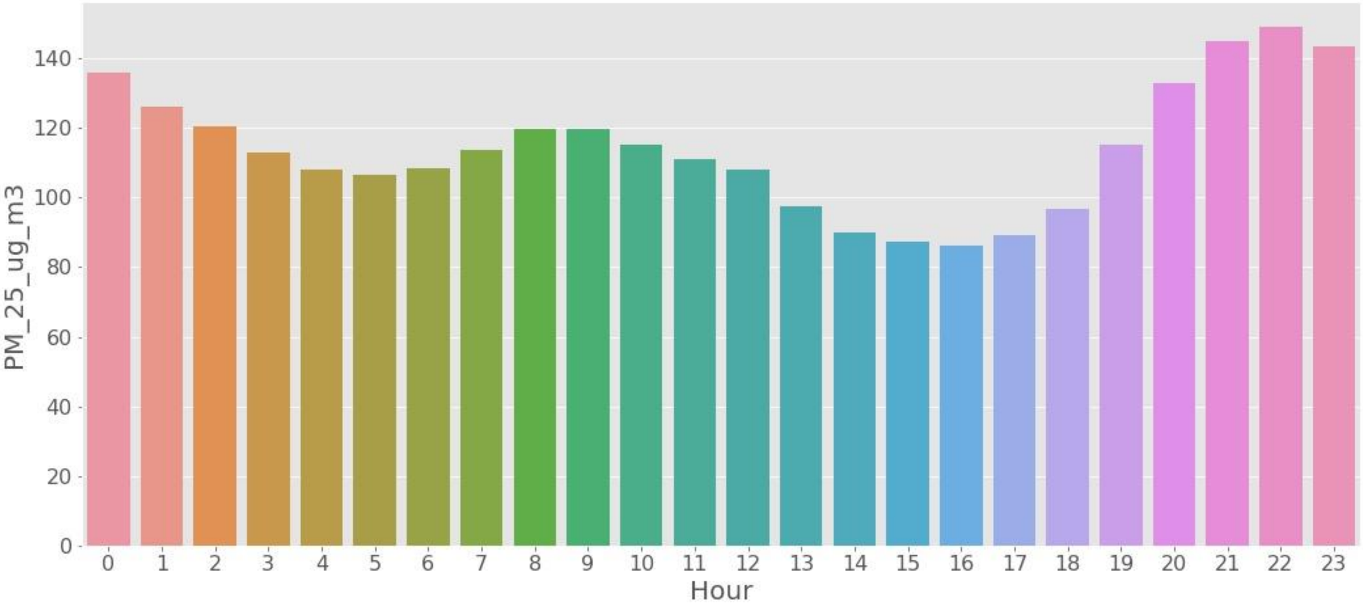

Figure 4: Bar Plot for average Hourly PM2.5 Values

To retain the cyclical nature of time-based features such as hour, day of week as well as month, they were transformed to create sine and cosine components. Using this approach, 11 pm and 12 am are placed close to each other as would be expected. The same logic was applied to create similar features for wind direction.

Further the correlation plot as seen below yielded the following observations:

- Negative correlations for temperature and wind speed with PM2.5
- Slight positive correlations with other pollutant features such as Sulphur dioxide, carbon monoxide and nitrogen-based compounds.
- Multi collinearity between some pollutant features such as carbon monoxide and nitrogen-based compounds as well between pairs of nitrogen-based compounds.

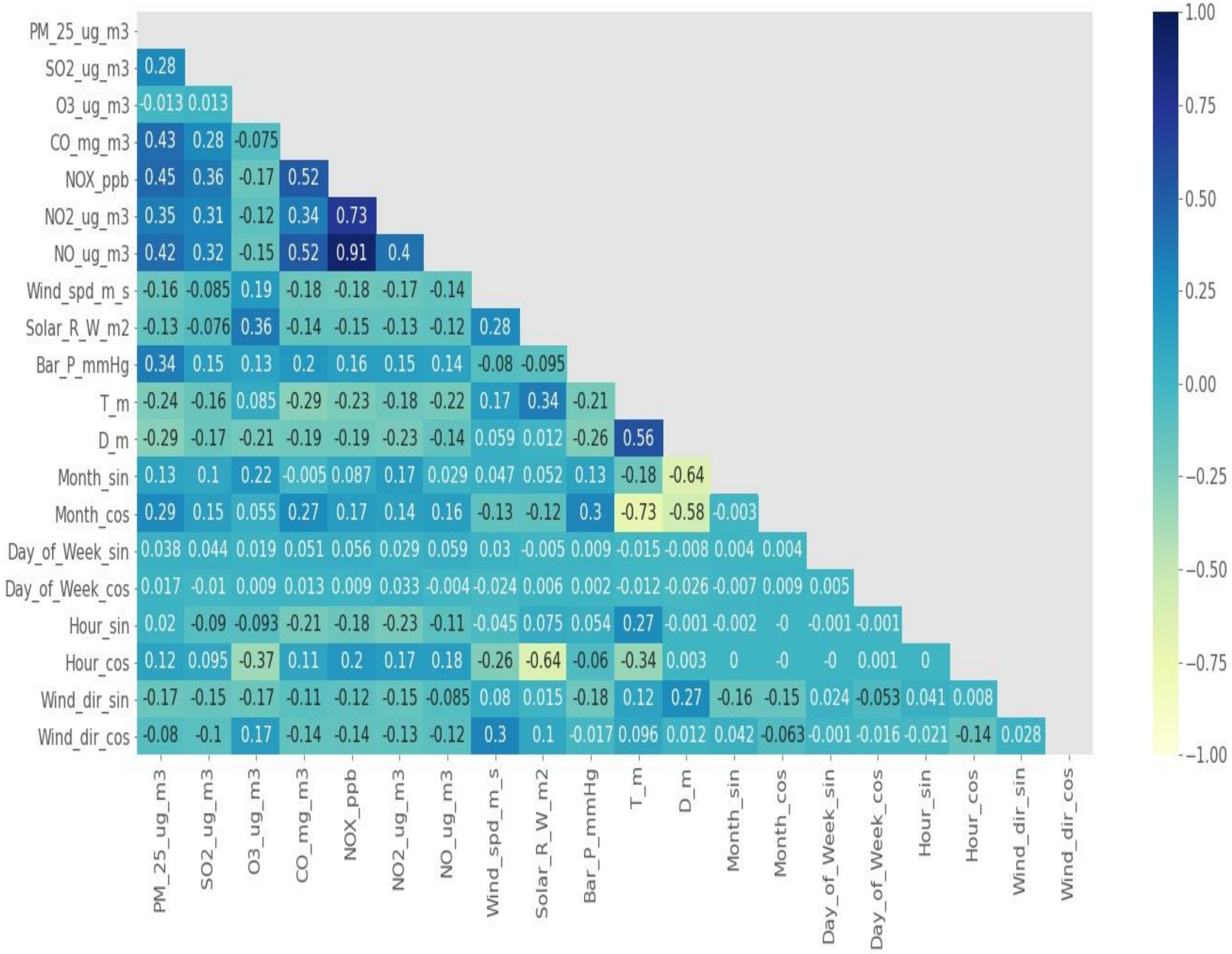

Figure 5: Correlation Heatmap for average all features

Modeling

As the dataset is a time series, regular methods such as splitting randomly into train-validation and test sets would not be a proper method to create the model as it is possible that the model gets trained on future values and is used to predict past values. Thus, it is necessary to sequentially split the data into train-validation and test sets. As a case study, models were created between a random split and time-based split for the sake of checking performance.

Also, previous time features or lag features are important features that add to the predictive power of a model. The random vs time based split case study also checked for modeling performance with and without such lag features.

The next consideration was to about cross-validating models. Regular cross-validation using random splits would not be applicable for a time series as mentioned above. Hence, I used a walk forward validation or forward chaining approach for the purposes of cross-validation. The schematic below[5] explains this approach where each subsequent test split is added to the training set to be used to predict the test set whose size remains constant. Multiple models equal to the number of splits need to be trained with this method. In addition, I used a nested cross-validation approach to tune hyper-parameters for the models as shown in the schematic[6] below.

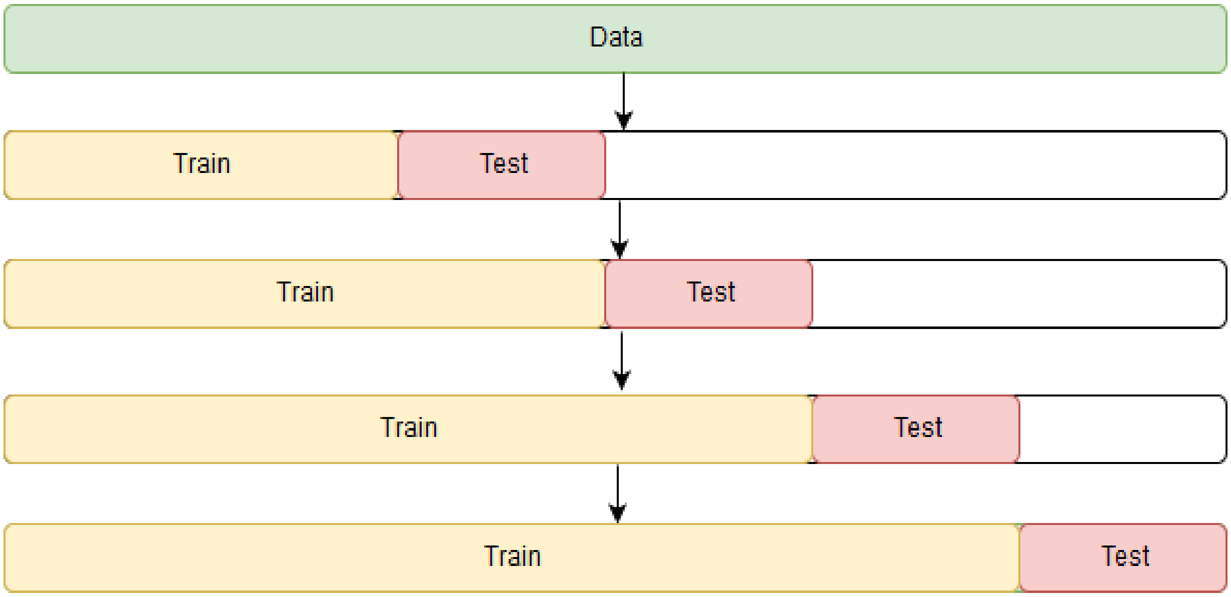

Figure 6: Forward Chaining or Walk Forward Validation

The walk forward approach avoids data leakage that can happen with a random split. With nested cross validation, the inner loop involves tuning hyperparameters using the given training split and later calculating average performance on the test splits in the outer loop. This provides a truer sense of model error rather than simply choosing an arbitrary test set to make predictions.

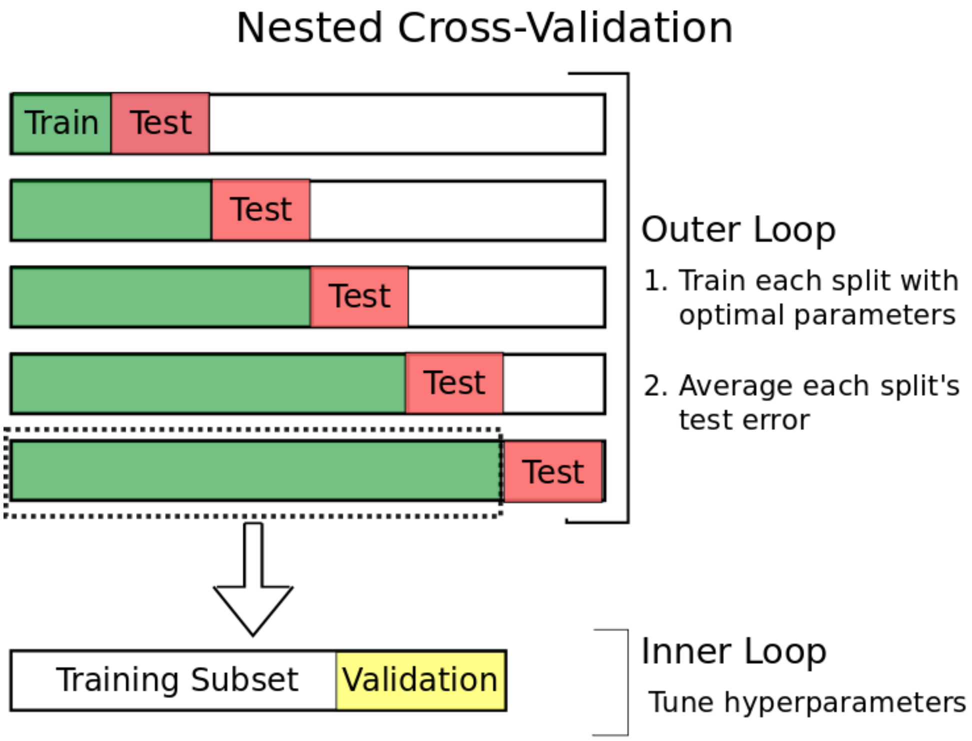

Figure 7: Nested Cross Validation

Bagging and boosting models were tested with the above-mentioned approaches. The models included Random Forest Regressor, Gradient Boosting Regressor, Light Gradient Boosting Regressor and XGBoosting Regressor.

For modeling tests between random vs time series split, all features specified in the data wrangling section were used. Light Gradient Boost was the only model used due to its fast training time. The results from these tests are shown in the following table:

Type of Split	Lag Features	Train RMSE	Test RMSE	Train MAE	Test MAE
Random Split	No	59.28	72.18	36.78	47.93
	Yes (3)	28.66	36.89	13.7	16.34
Time based Split	No	67.94	83.22	40.5	59.17
	Yes (3)	30.92	33.87	14.07	16.95

Table 1: Results for Random vs time-based splits

Following are some of the observations based on the results indicated above:

- Adding lag features significantly improve model performance irrespective of the type of split being used.
- Both random and time series split suffer from overfitting mainly for the case of no lags indicated by a large difference between training and testing metrics.
- The model based on random split performs better than time series split when no lags are added to the model whereas the latter slightly outperforms the former with addition of lag features.

All the models above were based on three train-test splits and 3 splits for nested cross validation.

While testing the forward chaining approach, features were checked sequentially to see if they improved model performance.
To start with, the number of lag features were tuned as a hyper-parameter.

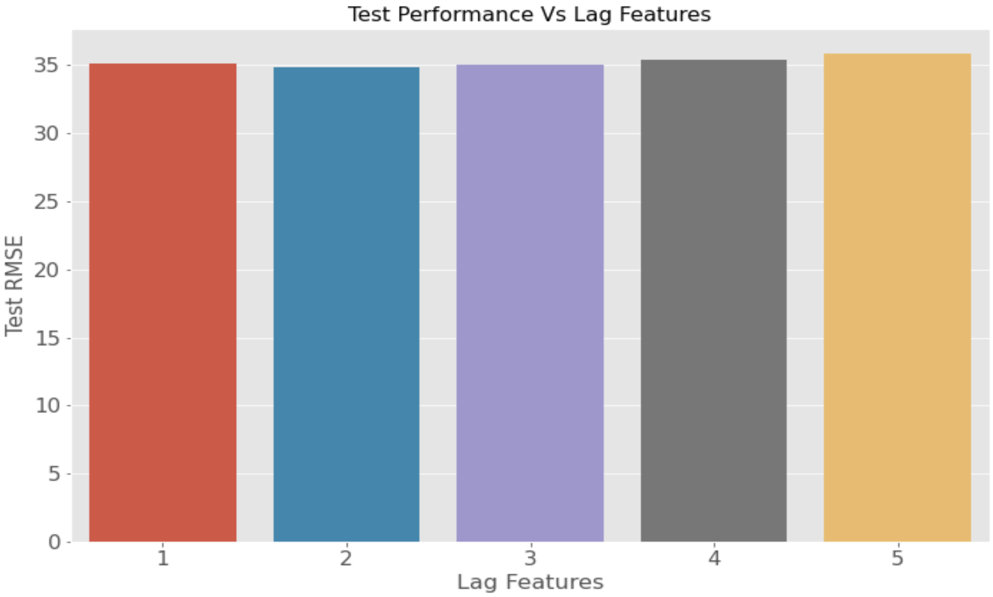

Figure 8: Bar Plot for test RMSE vs lag features

For example, the bar plot above for Light Gradient Boost with increasing lag features indicates that there is no significant improvement in performance beyond the first lag feature. This behavior was similar across all the tested models.

It was observed from modeling results that the first lag feature for PM2.5 in addition to the transformed time features mentioned above were sufficient to obtain the best performance irrespective of the type of model. Weather and other pollutant features did not add to the performance across all models.

The following table shows the results for the four models using a 3 split forward chaining approach and the first lag and time features as independent variables.

Model	Train RMSE	Test RMSE	Train MAE	Test MAE
Random Forest	37.49	33.56	14.82	15.66
Gradient Boost	38.62	34.35	17.39	16.62
Light Gradient Boost	38.6	33.85	16.88	16.22
XGBoost	38	34.05	16.68	16

Table 2: Model Performance for Walk Forward Validation

All models show similar performance with Random Forest slightly ahead of the boosting methods.

At a more granular level, forward chaining can also be used to make single step hourly forecasts instead of a small number of sequential splits. This approach though, can be very time consuming as a new model needs to be created for each of the new predictions[7]. The following figure shows results of making 200 single step hourly predictions using Light Gradient Boost. This model had a test set RMSE of 25.62 versus 35.14 for the training set. As seen in the mean feature importance, the first lag feature has the highest importance. Also, similar to the above table, training performance is poorer compared to the test set performance. This may be due to the smaller size of the testing set which remains constant compared to the training set which increases with each new model due to forward chaining.

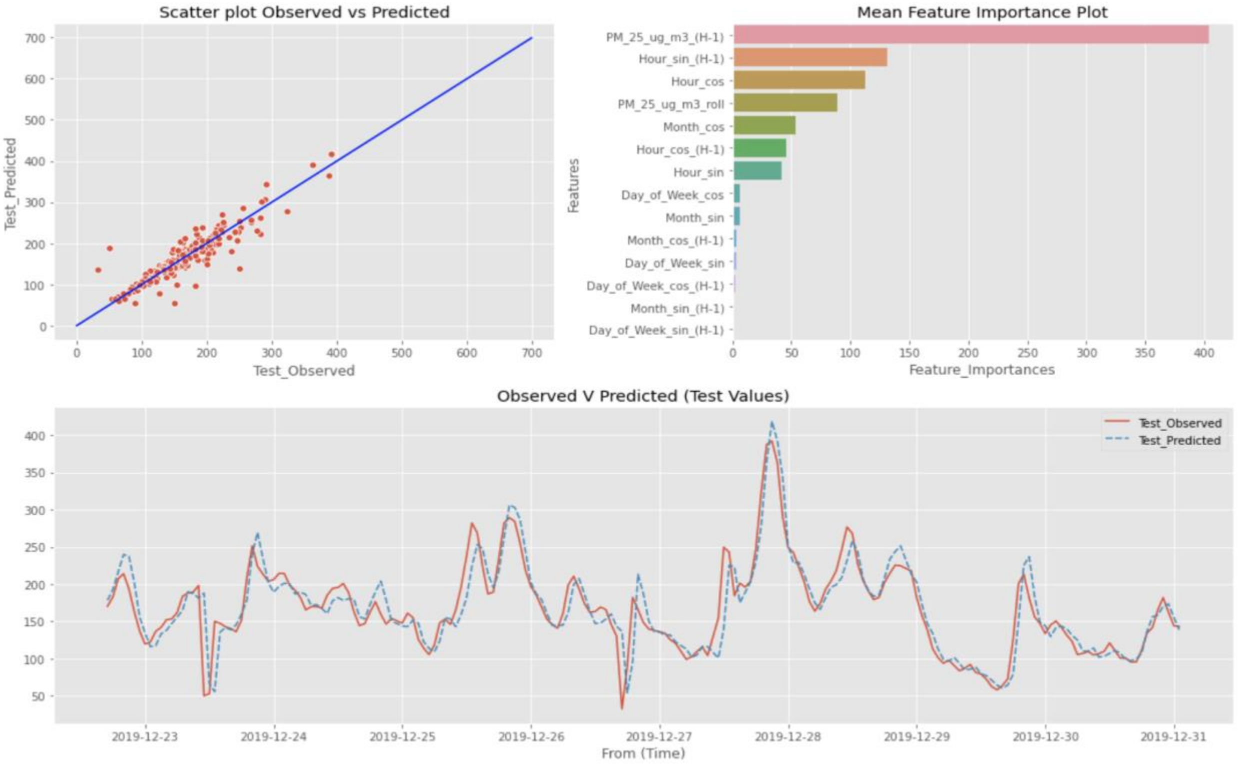

Figure 9: Hourly Predictions for Light Gradient Boost

Conclusions

This project attempted to predict PM2.5 by applied supervised learning to a multivariate time series.
For the tests between random vs time-based splits, models based on splitting randomly and having no lag features performed better compared to time-based splits whereas this behavior flipped on the addition of lag features.
Lag features significantly improved model performance.
For the forward chaining approach, features were tested systematically with the aim of removing unnecessary features. It was observed that the first lag feature was sufficient to improve model performance. Apart from a slight improvement with adding time-based features, weather and other pollutant features did not help improve the predictive power of the models.

For the studies above, nested cross-validation was used to tune hyper-parameters and avoid problems such as data leakage and selecting an arbitrary test set for the case of forward chaining.
All the models had similar performance metrics which showed that the right features contributed more than the type of the model.

Future Work

A different imputation and outlier treatment strategy can be followed to further clean the data based on domain knowledge to see if they contribute more to the modeling performance.
The work may also be extended further to create a real time tool that takes in updated data and creates forecasts for subsequent future time lines.

References

[1]https://www.theguardian.com/environment/2019/mar/12/air-pollution-deaths-are-double-previous-estimates-finds-research
[2] https://ourworldindata.org/air-pollution
[3]https://www.epa.gov/pm-pollution/particulate-matter-pm-basics#PM
[4]https://www.thelancet.com/journals/lanplh/article/PIIS2542-5196(20)30298-9/fulltext
[5]https://www.researchgate.net/publication/341618027_Forecasting_Sales_of_Truck_Components_A_Machine_Learning_Approach
[6]https://towardsdatascience.com/time-series-nested-cross-validation-76adba623eb9
[7]https://machinelearningmastery.com/backtest-machine-learning-models-time-series-forecasting
[8]https://medium.com/mongolian-data-stories/ulaanbaatar-air-pollution-part-1-35e17c83f70b
[9]https://machinelearningmastery.com/convert-time-series-supervised-learning-problem-python/