Beijing Air Quality Prediction and Analysis

Chen Zhe (A0091882W), Gao Xiongyi (A0212206Y), Luan Min (A0105743L),
Qiao Wei (A0105572L), Zhang Hao (A0210996X)

Introduction

With increasing concerns about the environment and living conditions nowadays, the air quality index, which gives people vital information about the air condition in their living place and how air quality in cities impacts public health, causes people's great attention. It is useful and meaningful if we could predict the air quality index for the future based on gathered historical meteorological data. PM2.5, an index measures atmospheric particulate matter (PM) that have a diameter of fewer than 2.5 micrometers, which may trigger or worsen chronic diseases such as asthma, heart attack, bronchitis, and other respiratory problems. It becomes one of the most important indices, which measures air quality and draws much concern from the public. Therefore, in this project, we aim to explore and build a prediction model of air quality index PM2.5 with the meteorological dataset including temperature, pressure, wind, rain, etc. at multiple sites of Beijing by applying different methods, including linear regression, support vector regression, decision tree, neural networks and time series model.

We used ***Beijing Multi-Site Air-Quality Data Set*** [1] from Kaggle. It includes hourly air pollutants data from 12 Chinese nationally-controlled air-quality monitoring stations: Aotizhongxin, Changping, Dingling, Dongsi, Guanyuan, Gucheng, Huairou, Nongzhanguan, Shunyi, Tiantan, Wanliu, Wanshouxigong. And the air-quality data is from the Beijing Municipal Environmental Monitoring Center. The meteorological data in each air-quality site are matched with the nearest weather station from the China Meteorological Administration. The period is from March 1st, 2013 to February 28th, 2017 with an hour as the smallest granularity. For the air pollutant data, it contains hourly concentrations of 6 main air pollutants, including PM2.5, PM10, SO2, NO2, CO, O3 with the unit of ug/m^3. For the air-quality data, it contains hourly information of 6 relevant meteorological variables, including temperature(TEMP)(degree Celsius), pressure(PRES)(hPa), dew point temperature(DEWP)(degree Celsius), rain precipitation (RAIN)(mm), wind direction and wind speed(WSPM) (m/s).

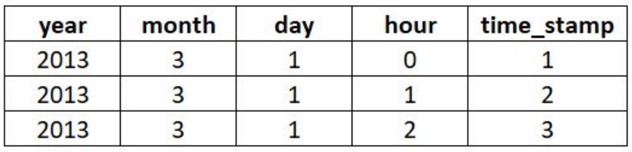
Preliminary analysis

Data Split

To balance data from 12 different stations, we split the raw data of each station into 85% training data and 15% test data and combine training and test datasets from all stations into one training dataset and one test dataset. The training dataset will then be split into training and cross-validation dataset according to the needs of each model. For a consistent comparison of different models' prediction performance, we evaluate the models by the same test dataset. Besides, as the time series model uses historical data to predict future values, the training dataset is data of older years and the test dataset is a more recent period.

Data Transformation

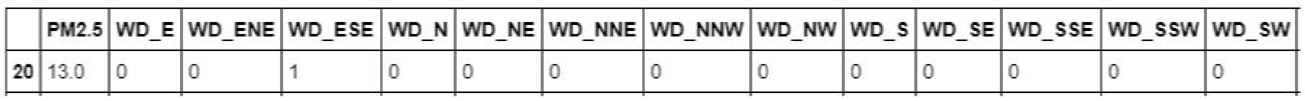
The time information of the raw dataset is separated into 4 columns namely year, month, day, and hour. To easily sort time and identify air condition change with time, we convert each row of year, month, day, and hour data into one numerical value in a new column "time_stamp".



year	month	day	hour	time_stamp
2013	3	1	0	1
2013	3	1	1	2
2013	3	1	2	3

Figure 1. Time_stamp Transformation

As the wind direction is categorical data with 16 categories (WD_E WD_ENE, WD_ESE, WD_N, WD_NE, WD_NNE, WD_NNW, WD_NW, WD_S, WD_SE, WD_SSE, WD_SSW, WD_SW, WD_W, WD_WNW, WD_WSW), we convert this categorical feature into 16 dummy binary variables to construct the model as below.



	PM2.5	WD_E	WD_ENE	WD_ESE	WD_N	WD_NE	WD_NNE	WD_NNW	WD_NW	WD_S	WD_SE	WD_SSE	WD_SSW	WD_SW
20	13.0	0	0	1	0	0	0	0	0	0	0	0	0	0

Figure 2. Applying Dummy Variables

Data Cleaning

The original dataset has 420,768 rows of which 38,600 rows include at least one missing data. We remove those rows with missing data considering the impact is negligible in a large dataset. Besides, as the time series model's predictions depend a lot on previous records, replacing missing data with the mean or median of the dataset will likely reduce the accuracy of the time series model, thus we chose to remove missing data.

Data Normalization

From the box plot of the features under the same coordinate system, we observe the scales of different features vary a lot, for example, the CO max value is much larger than other pollutants. The drastically different scales will make the feature with a larger scale completely dominate other features when we train the prediction model. Therefore, it is necessary to normalize different features on a common scale.

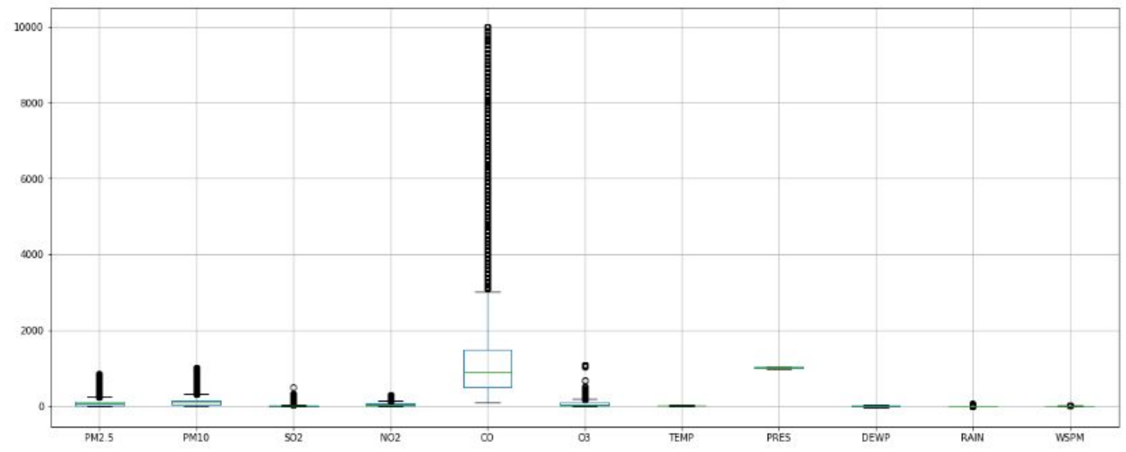


Figure 3. Box Plot Without Normalization

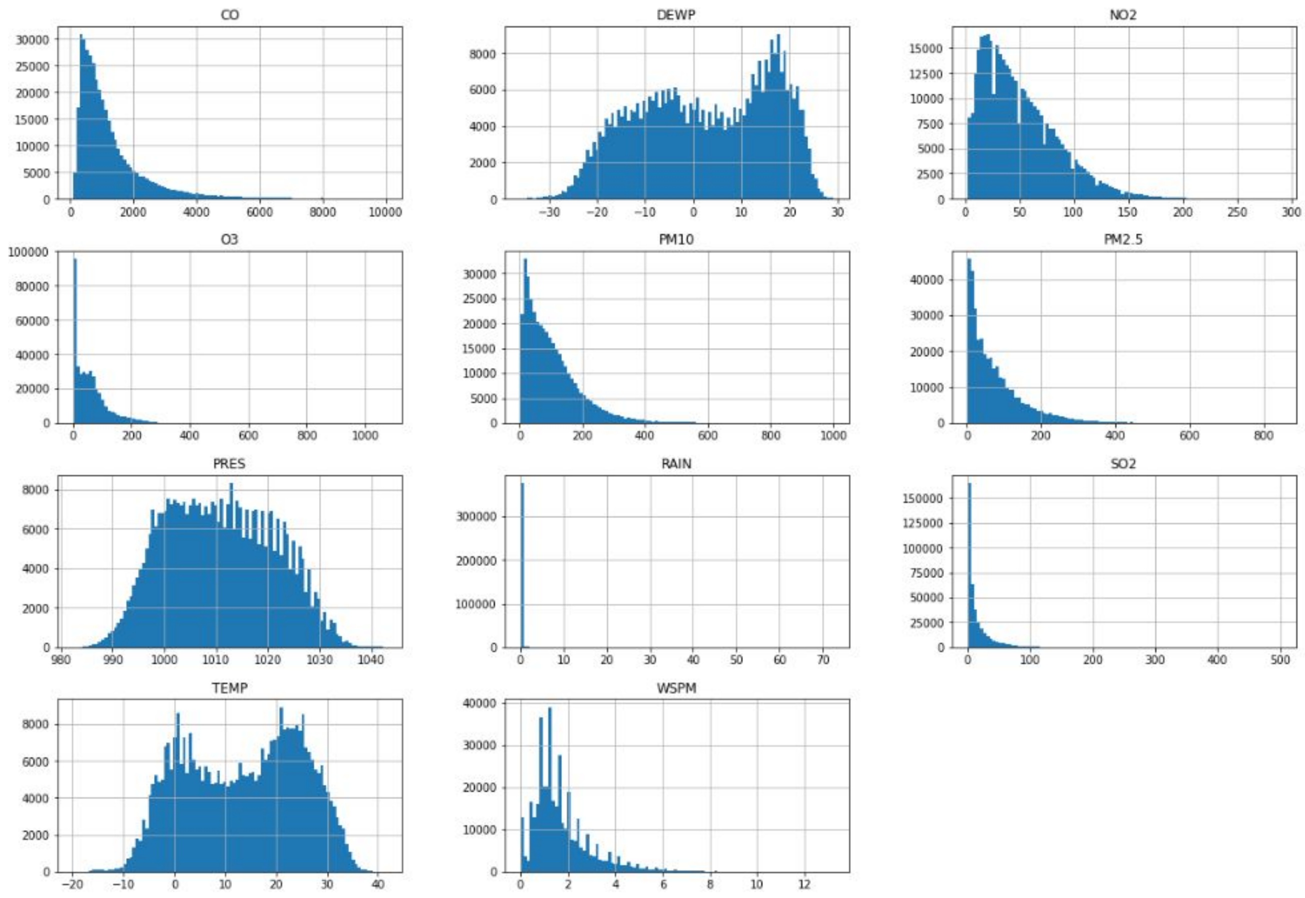


Figure 4. Histogram Without Normalization

We use Min-max normalization approach to normalize data of 10 features 'PM10', 'SO2', 'NO2', 'CO', 'O3', 'TEMP', 'PRES', 'DEWP', 'RAIN' and 'WSPM'. For every feature, the minimum value is normalized to 0, the maximum value is normalized to 1, and every other value is transformed into a value between 0 and 1 using the formula: (value-min)/(max-min). After Normalization, we get the box plot with features on a common scale as below.

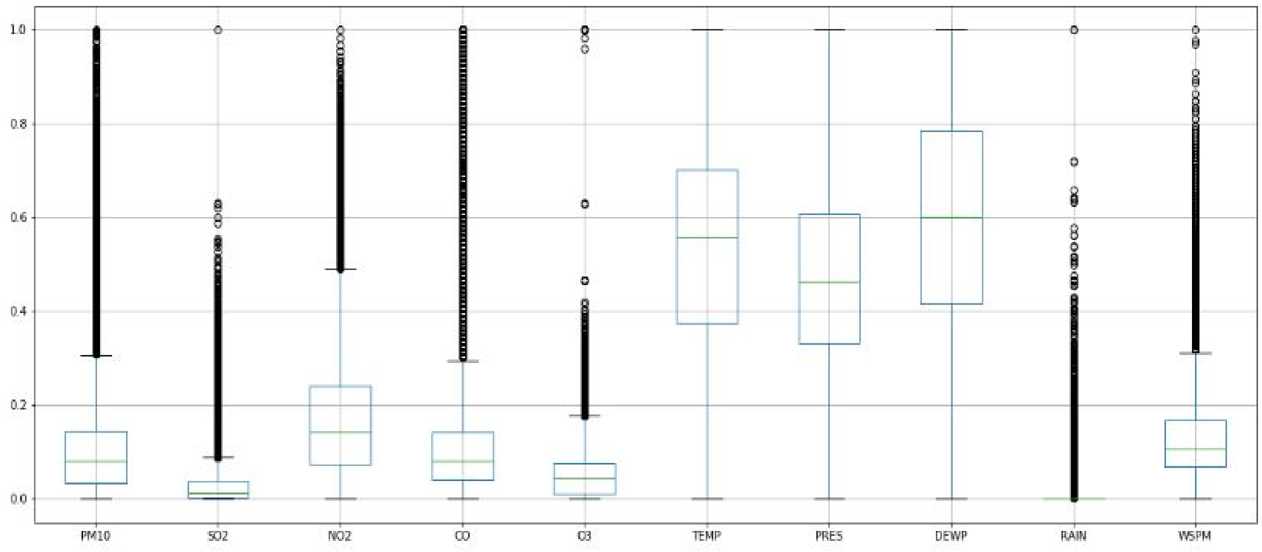
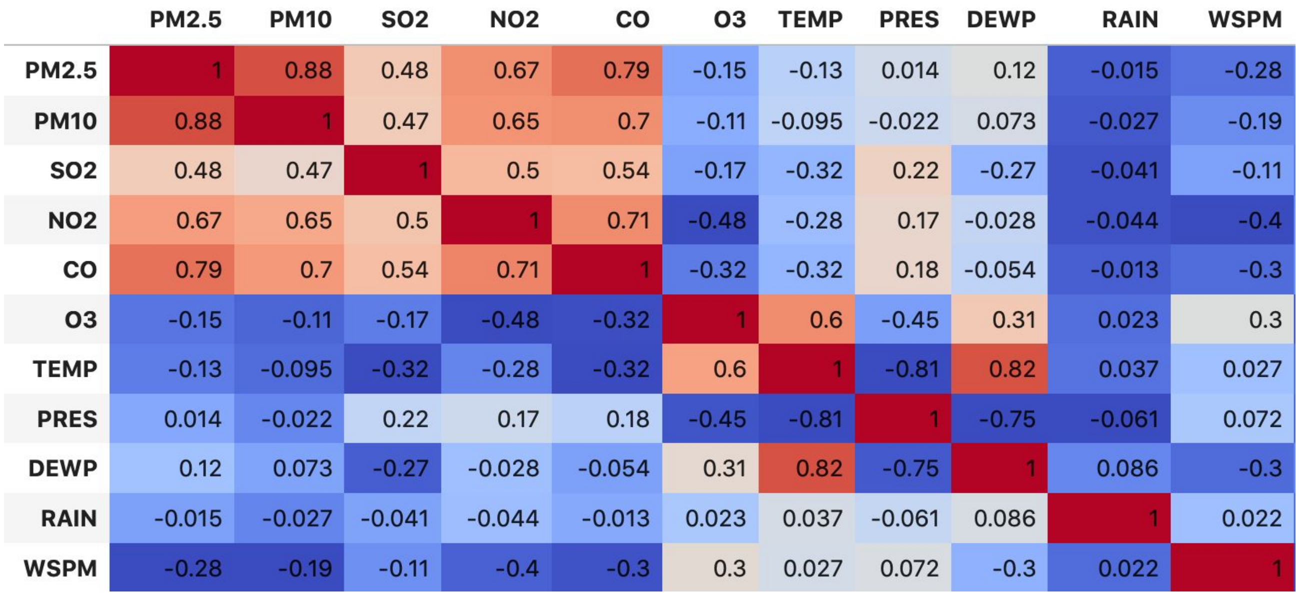


Figure 5. Box Plot After Normalization

Data Correlation Analysis

Through plotting the pairwise relationships and calculating the correlation coefficient of air pollutants data and air quality data, we find some interesting relationships among different data, which is useful for our prediction and give us enlightenment on building the model.



	PM2.5	PM10	SO2	NO2	CO	O3	TEMP	PRES	DEWP	RAIN	WSPM
PM2.5	1	0.88	0.48	0.67	0.79	-0.15	-0.13	0.014	0.12	-0.015	-0.28
PM10	0.88	1	0.47	0.65	0.7	-0.11	-0.095	-0.022	0.073	-0.027	-0.19
SO2	0.48	0.47	1	0.5	0.54	-0.17	-0.32	0.22	-0.27	-0.041	-0.11
NO2	0.67	0.65	0.5	1	0.71	-0.48	-0.28	0.17	-0.028	-0.044	-0.4
CO	0.79	0.7	0.54	0.71	1	-0.32	-0.32	0.18	-0.054	-0.013	-0.3
O3	-0.15	-0.11	-0.17	-0.48	-0.32	1	0.6	-0.45	0.31	0.023	0.3
TEMP	-0.13	-0.095	-0.32	-0.28	-0.32	0.6	1	-0.81	0.82	0.037	0.027
PRES	0.014	-0.022	0.22	0.17	0.18	-0.45	-0.81	1	-0.75	-0.061	0.072
DEWP	0.12	0.073	-0.27	-0.028	-0.054	0.31	0.82	-0.75	1	0.086	-0.3
RAIN	-0.015	-0.027	-0.041	-0.044	-0.013	0.023	0.037	-0.061	0.086	1	0.022
WSPM	-0.28	-0.19	-0.11	-0.4	-0.3	0.3	0.027	0.072	-0.3	0.022	1

Figure 6. Correlation Coefficient Heat Map

From the values of the correlation coefficient heat map in Figure 6, we observe that concentration of PM2.5 has a strong positive linear relationship with the concentration of PM10 and CO; a moderate positive linear relationship with the concentration of NO2 and SO2. Although it has a weak linear relationship with the value of rain and wind speed (WSPM), we can observe the negative relationship between them, which follows common sense that rain and wind help clear away pollutants in the air.

For the relationship between PM2.5 and wind direction, we observe that different wind directions may have a positive or negative influence on PM2.5, although the relationship is not very strong. It is possibly due to air taken to Beijing by the wind blowing from different directions outside the Beijing area is quite different in air quality. Especially for the wind from the direction of NW, NNW, and WNW as in Table 7 below, it has a relatively strong negative influence on PM2.5 compared to other directions, as we know the northwest direction of Beijing is a mountain area with better air quality.

Wind Direction	Correlation Coefficient
WD_E	0.0751
WD_ESE	0.0689
WD_ENE	0.0670
WD_SE	0.0469
WD_SSE	0.0373
WD_S	0.0270
WD_NE	0.0250
WD_SSW	0.0140
WD_SW	-0.0005
WD_WSW	-0.0062
WD_W	-0.0139
WD_NNE	-0.0305
WD_N	-0.0447
WD_WNW	-0.0720
WD_NNW	-0.0833
WD_NW	-0.1025

Table 7. The correlation coefficient between wind direction and PM2.5

Evaluation Metrics

We use Root Mean Squared Error(RMSE), Mean Absolute Error(MAE), and Mean Squared Error(MSE) to evaluate the accuracy of prediction models and make comparisons on the performance of different models. We also include several different measurements of accuracy: ”±10/±20/±50/±100 Acc” denotes the percentage of prediction results that fall into the interval of ±10/20/50/100 of the real value.

Regression Approach

Variables Dependency

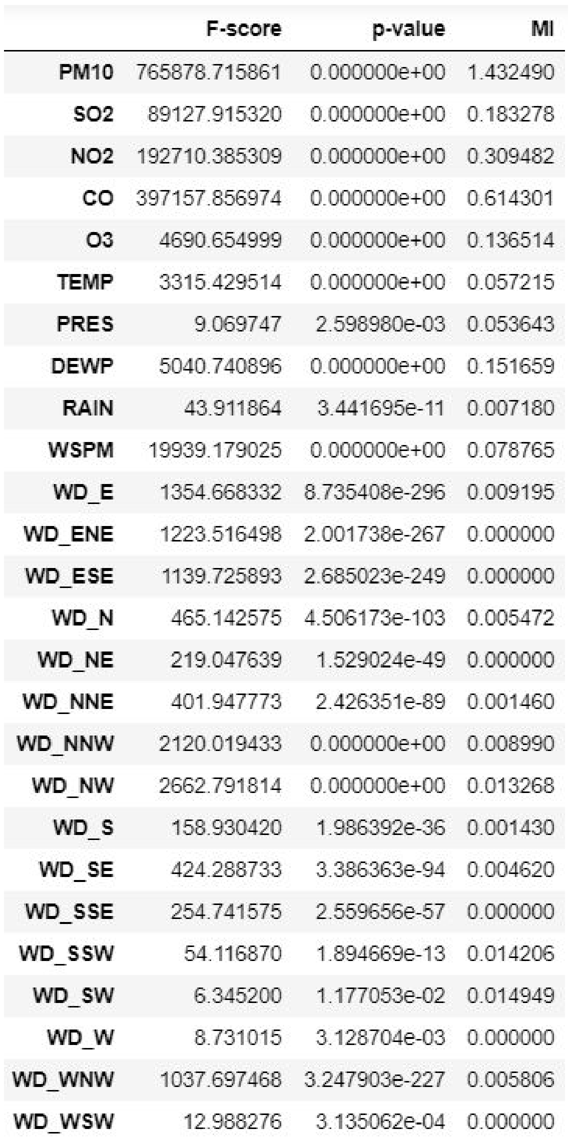
To explore the variables dependency and linear relationship between PM2.5 and other features, we calculate F-score, p-value, and Mutual Information value for feature selection.

F test is a statistical test used to compare model X created without the feature and model Y with the feature. It does a hypothesis testing and under null hypothesis, model Y does not provide a significantly better fit than model X. The null hypothesis is rejected if F-score is large. So a higher F-score implies a highly correlated feature and a lower F-score implies a less correlated feature.

Similar to F-score, p-value tests the null hypothesis that the input feature has no correlation with the target attribute. If the p-value is small enough and less than the significance level, it means sample data has enough evidence to reject the null hypothesis and that feature is a significant input attribute.

Mutual information measures the amount of information obtained for the target attribute by observing the input feature. Its value equals to zero if and only if the input attribute and target attribute are independent, and higher values mean higher dependency.

Through the p-value, mutual information, we observe that these 3 values show similar estimation results of the linear relationship between the dependent variable(PM2.5) and other independent variables(other air pollutants data except PM2.5 and air quality data). Dependent variable value has a strong linear relationship with PM10 and CO since PM10 and CO get the largest two F score and Mutual Information(MI) value and zero value of p-value. For other independent variables like SO2, NO2, O3, TEMP, DEWP, WSPM, WD_NNW, and WD_NW, they have a moderate linear relationship with PM2.5 as their p-value is zero and their F score is not very large. For the remaining air condition data like PRES, RAIN, and other wind directions, they have a relatively smaller F score, MI value, and relatively larger p-value, which indicates a weak linear relationship with PM2.5 value.



	F-score	p-value	MI
PM10	765878.715861	0.000000e+00	1.432490
SO2	89127.915320	0.000000e+00	0.183278
NO2	192710.385309	0.000000e+00	0.309482
CO	397157.856974	0.000000e+00	0.614301
O3	4690.654999	0.000000e+00	0.136514
TEMP	3315.429514	0.000000e+00	0.057215
PRES	9.069747	2.598980e-03	0.053643
DEWP	5040.740896	0.000000e+00	0.151659
RAIN	43.911864	3.441695e-11	0.007180
WSPM	19939.179025	0.000000e+00	0.078765
WD_E	1354.668332	8.735408e-296	0.009195
WD_ENE	1223.516498	2.001738e-267	0.000000
WD_ESE	1139.725893	2.685023e-249	0.000000
WD_N	465.142575	4.506173e-103	0.005472
WD_NE	219.047639	1.529024e-49	0.000000
WD_NNE	401.947773	2.426351e-89	0.001460
WD_NNW	2120.019433	0.000000e+00	0.008990
WD_NW	2662.791814	0.000000e+00	0.013268
WD_S	158.930420	1.986392e-36	0.001430
WD_SE	424.288733	3.386363e-94	0.004620
WD_SSE	254.741575	2.559656e-57	0.000000
WD_SSW	54.116870	1.894669e-13	0.014206
WD_SW	6.345200	1.177053e-02	0.014949
WD_W	8.731015	3.128704e-03	0.000000
WD_WNW	1037.697468	3.247903e-227	0.005806
WD_WSW	12.988276	3.135062e-04	0.000000

Table 8. F-score, P-value and Mutual Information results

Linear Regression

Linear regression is a linear approach to modeling the relationship between a dependent variable and one or more independent variables. We regard PM2.5 as a dependent variable to be predicted and other air condition data as independent variables to help prediction, and construct the linear regression model which minimizes the residual sum of squares between the independent variables and target predicted.

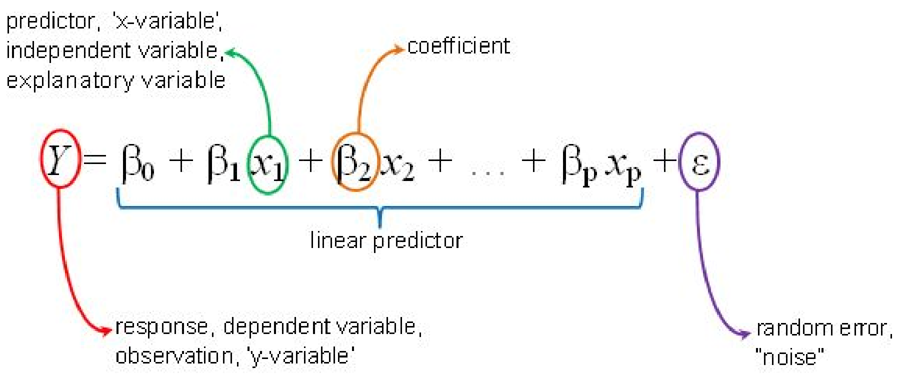
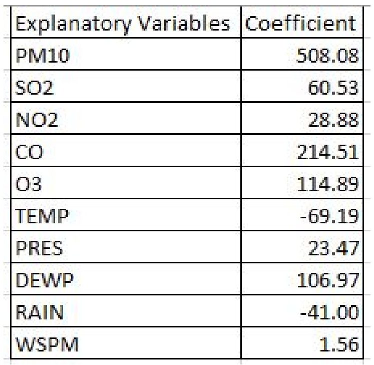

predictor, 'x-variable', independent variable, explanatory variable
coefficient
$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p + \varepsilon$
linear predictor
response, dependent variable, observation, 'y-variable'
random error, "noise"

Figure 9. Multiple Linear Regression[5]

Table 10. is the coefficient of different independent variables in our linear regression. From the absolute value of the coefficient, we can observe that the PM10 and CO have the largest two coefficients, which matches the linear relationship we found through F score and p-value. The value of these 2 inputs has the strongest positive effect on the prediction result of PM2.5.

For other inputs' coefficients with smaller values, they imply a relatively weak relationship with the value of PM2.5. SO2, NO2, CO, O3, PRES, DEWP positively related to the value of PM2.5, whereas TEMP and RAIN negatively related to the value of PM2.5, which is consistent with what we find in the data correlation analysis.

For WSPM(Wind Speed), it has the smallest positive value of the coefficient, but not negatively related to PM2.5. It is because the wind from different directions may have a totally different relationship with the value of PM2.5. As the WSPM input does not include the information of wind direction, after aggregation of effects of different wind directions, it causes WSPM(Wind Speed) with a small value of the coefficient. For WD(Wind Direction), as the data of different wind directions are in binary format and linear regression has its limitation to handle binary data, we do not take the wind direction information into consideration for the linear regression model. However, for other models below, we take WD(Wind Direction) data into consideration.



Explanatory Variables	Coefficient
PM10	508.08
SO2	60.53
NO2	28.88
CO	214.51
O3	114.89
TEMP	-69.19
PRES	23.47
DEWP	106.97
RAIN	-41.00
WSPM	1.56

Table 10. Coefficients From Linear Regression

Although the wind speed information does not help much on the prediction of PM2.5, the other air condition information provides enough useful information, which helps with prediction. Below is the result we get from the linear regression model.

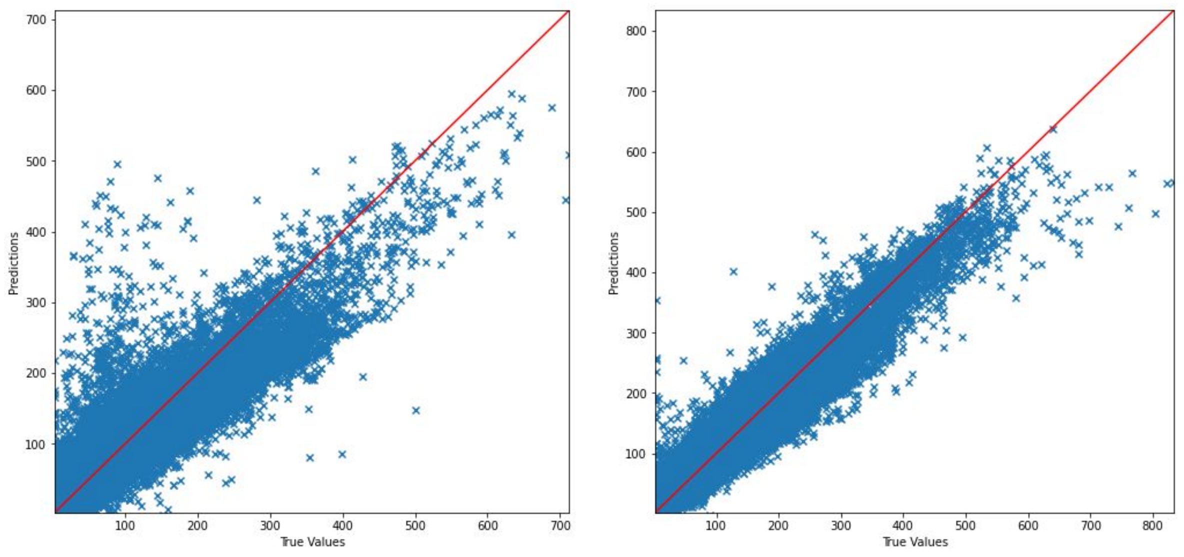


Figure 11. True PM2.5 vs Predicted PM2.5 on ValidationSet(left) and TestingSet(right)

Model Name	MAE	RMSE	MSE	±10 Acc	±20 Acc	±50 Acc	±100 Acc
Linear Regression	17.68	30.68	642.31	42.07%	69.26%	94.08%	99.58%

Table 12. Linear Regression Metrics

As the linear regression has the limitation that it cannot take the wind direction effect into consideration correctly, we try to explore other methods for better prediction of PM2.5.

Linear SVM Regression

Support Vector Machine maps data to a high-dimensional feature space such that data points could be categorized. It constructs a line or a hyperplane as the optimal boundary which separates the data into classes.
Similar to the principle of the support vector machine, the Linear Support Vector Regression also constructs a boundary that gives as wide a margin as possible between the data points that are closest to the support vectors. However, the goal of the support vector regression is different from the support vector machine. The algorithm does not care about the data points lying within a certain margin around the hyperplane, as they do not incur any cost. And the margin is defined by the parameter ϵ, which is set by users. As a linear regression algorithm, we try to take it to predict the PM2.5 and compare its performance with other prediction models.

$$y = \sum_{i=1}^{N} \left(\alpha_i - \alpha_i^*\right) \cdot \left\langle x_i, x \right\rangle + b$$

Below are the parameters used in training the model to predict PM2.5 based on the 26 transformed input variables. We set ϵ in the epsilon-insensitive loss function as 0, the tolerance for stopping criteria to be 0.0001, and use the epsilon-insensitive loss as the loss function.

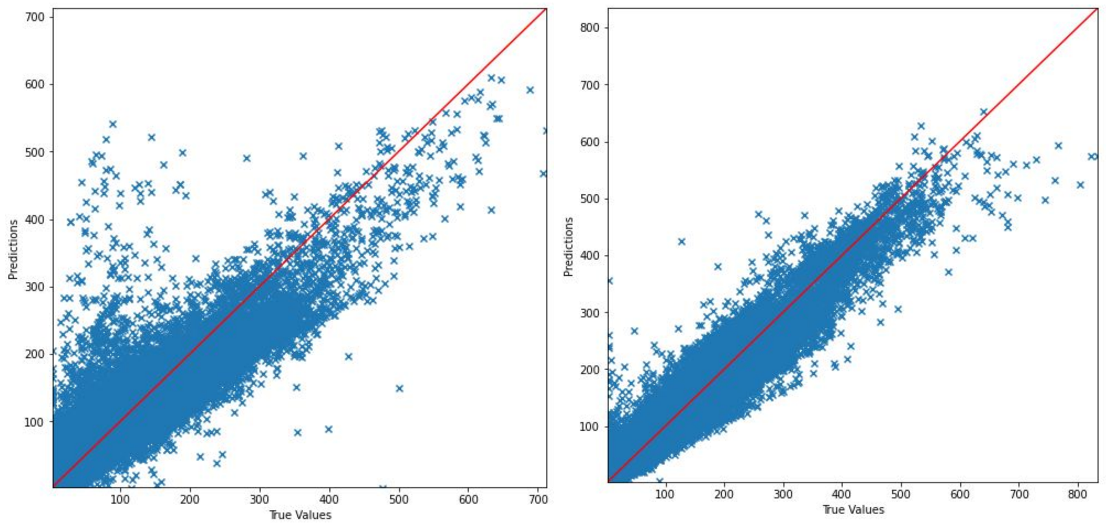


FIgure 13. True PM2.5 vs Predicted PM2.5 on ValidationSet(left) and TestingSet(right)

Model Name	MAE	RMSE	MSE	±10 Acc	±20 Acc	±50 Acc	±100 Acc
Linear Support Vector Regression	16.59	24.76	613.14	48.88%	71.63%	94.16%	99.59%

Figure 14. Linear Regression Metrics

XGBoost Regression

XGBoost is an ensemble of decision trees that uses a Gradient Boost framework, which minimizes errors when constructing multiple sequential trees. It has proven good results on small-to-medium tabular data. XGBoost was developed by Tianqi Chen and Carlos Guestrin for a research project at the University of Washington[2]. The advantage of XGBoost is the regularization techniques to reduce the generalization error, the built-in feature of missing data handling. Despite the complexity of various optimization components, it can be trained with less time compared to other complex tree-based models. XGBoost was implemented with parallelization, cache awareness and tree pruning with a max_depth parameter to enhance its training speed.

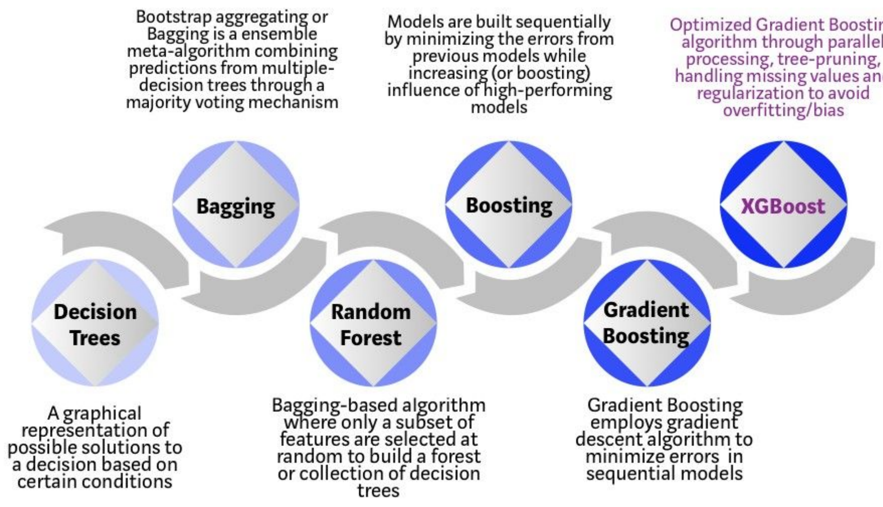
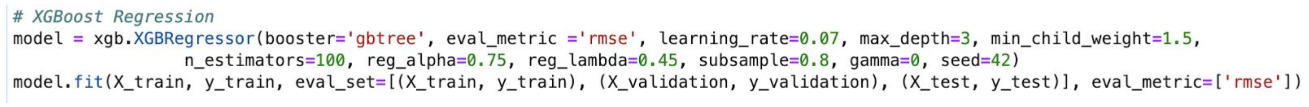

Bootstrap aggregating or Bagging is a ensemble meta-algorithm combining predictions from multiple-decision trees through a majority voting mechanism
Models are built sequentially by minimizing the errors from previous models while increasing (or boosting) influence of high-performing models
Optimized Gradient Boosting algorithm through parallel processing, tree-pruning, handling missing values and regularization to avoid overfitting/bias
Bagging
Boosting
XGBoost
Decision Trees
Random Forest
Gradient Boosting
A graphical representation of possible solutions to a decision based on certain conditions
Bagging-based algorithm where only a subset of features are selected at random to build a forest or collection of decision trees
Gradient Boosting employs gradient descent algorithm to minimize errors in sequential models

Figure 15. Evolution of XGBoost Algorithm[2]

Below are the parameters used in training the model to predict PM2.5 based on the 26 transformed input variables. GridSearch was used to explore the best combination of a given range of parameter sets.



```
# XGBoost Regression
model = xgb.XGBRegressor(booster='gbtree', eval_metric ='rmse', learning_rate=0.07, max_depth=3, min_child_weight=1.5,
                         n_estimators=100, reg_alpha=0.75, reg_lambda=0.45, subsample=0.8, gamma=0, seed=42)
model.fit(X_train, y_train, eval_set=[(X_train, y_train), (X_validation, y_validation), (X_test, y_test)], eval_metric=['rmse'])
```

Figure 16. Parameters of XGBoost

Due to the complexity of the algorithm, the time taken for training with XGBoost is about 44s, while it only took the basic Decision Tree about 2.67s. However, XGBoost has reduced the RMSE on the testing dataset by **25.5%** compared with basic Decision Tree prediction.

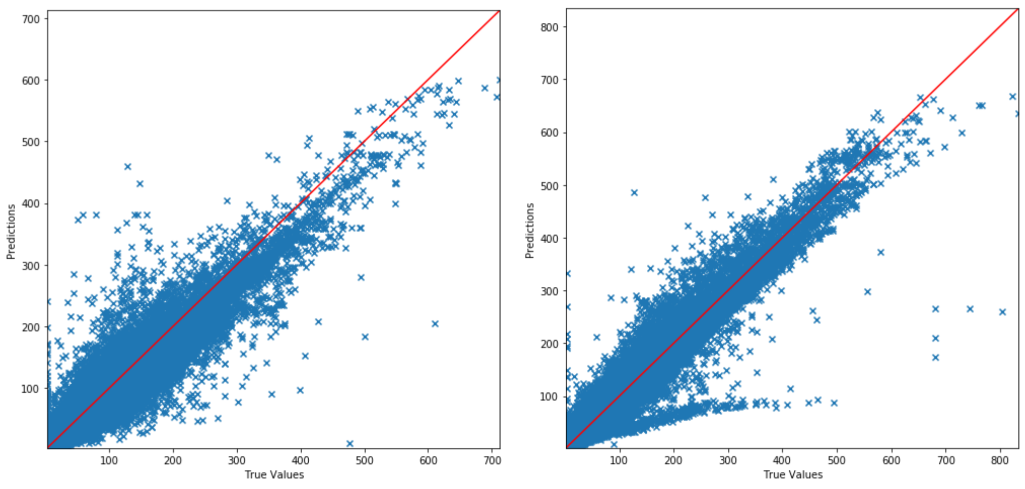


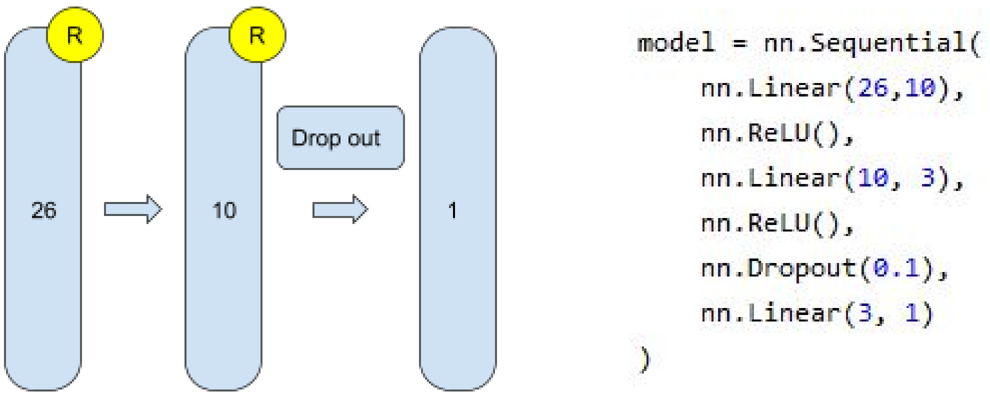
Figure 17. True PM2.5 vs Predicted PM2.5 on ValidationSet(left) and Testing Set(right)

Neural Network Approach

Besides the regression approach, we also tried the neural network methodology to predict the PM 2.5 value.

MLP Model

The first model we designed is a multi-level perceptron model with three dense layers connected together.



```
model = nn.Sequential(
    nn.Linear(26,10),
    nn.ReLU(),
    nn.Linear(10, 3),
    nn.ReLU(),
    nn.Dropout(0.1),
    nn.Linear(3, 1)
)
```

Figure 18. MLP Model Illustration (left) and code (right)

The first layer has 26 neurons to match the input dimension, fully connected by the hidden layer with 10 neurons. After the hidden layer, a drop out layer was introduced for the generalization purpose with a 10% drop out rate. Finally, it will generate the output layer which is a single value as the prediction for PM 2.5 value. The activation method we used between layers is the ReLU method.

Because the MLP model is not time-related, we shuffled all the training data and randomly split out 20% data as cross-validation data. Here are the other hyper-parameters we used for training:

Optimizer	Learning rate	Loss function	Batch size
Adam	1e-3	MSE	300

We trained this model for 80 epochs and recorded the validation result after each epoch.

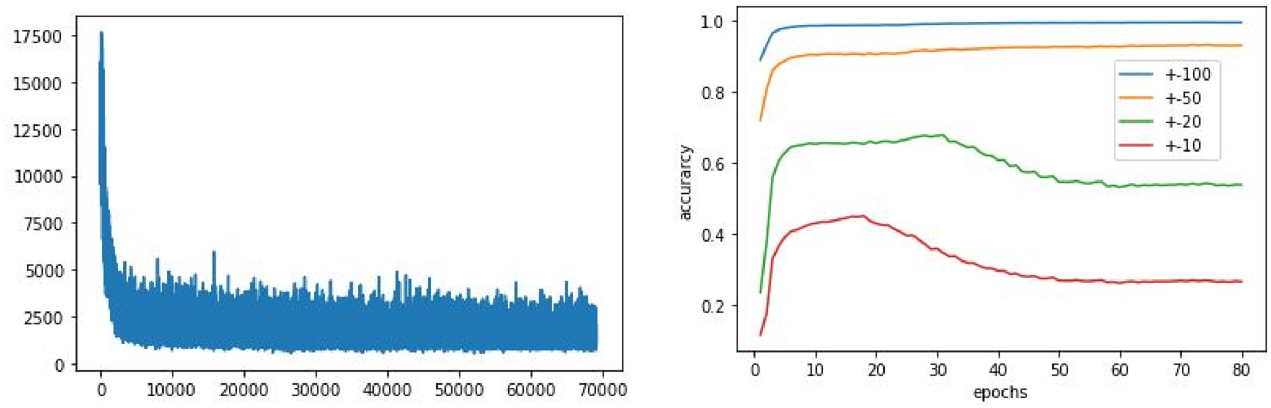


Figure 19. MLP Training loss value by batch (left) and validation accuracy (right)

Adam is an advanced stochastic gradient descent method, it will randomly select batch-size data from the training dataset in each step. So the loss value will shake up and down during the training process. Overall, we can see the loss value has converged well after several epochs. But from the validation accuracy result, the model seems to overfit and the accuracy

for ±10 and ±20 have a clear drop in the graph. Therefore, we chose the model parameters at 20 epochs as our final prediction model to prevent overfitting.
Here is the prediction result using test data.

MAE	RMSE	MSE	±10 Acc	±20 Acc	±50 Acc	±100 Acc
20.64	31.49	999.88	42.90%	65.50%	90.30%	98.40%

Table 20. MLP test result

LSTM Model

To further improve prediction accuracy, we want to try a time-related neural network. So after MLP, we created a Long-Short-Term-Memory model. LSTM is an advanced recurrent neural network.

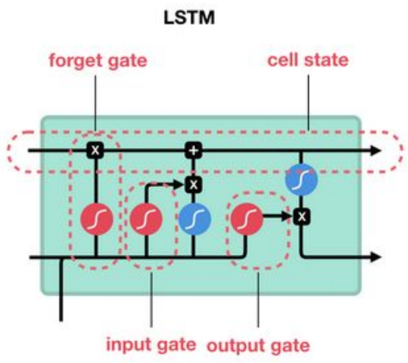


Figure 21. LSTM gates illustration

Compared with simple RNN, it introduced the forget gate, cell gate, input gate, and output gate to control the update and forget of the value from the previous cell.
In our model, we will feed in 48 hours of data as input and predict the last 24 hours PM 2.5 value.

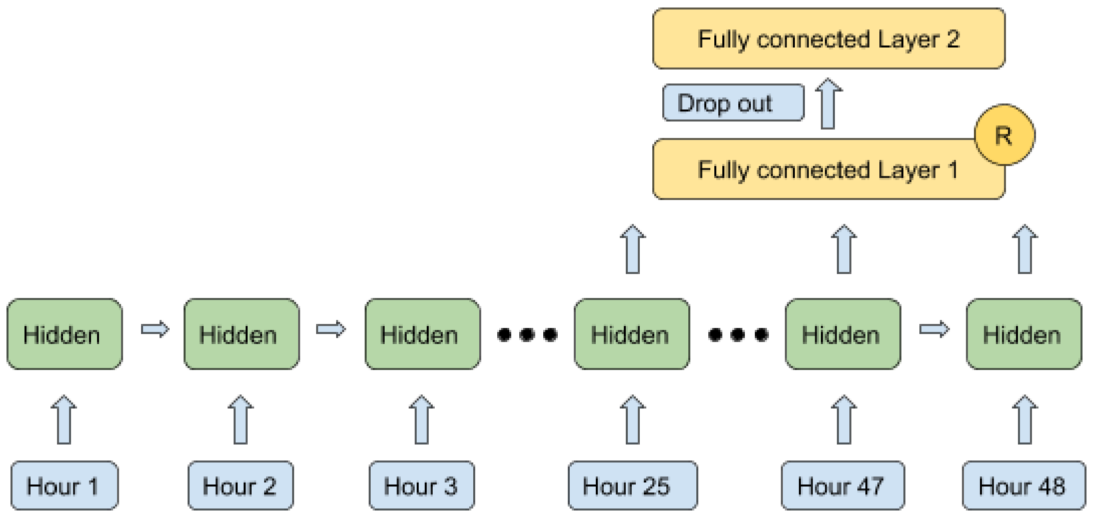
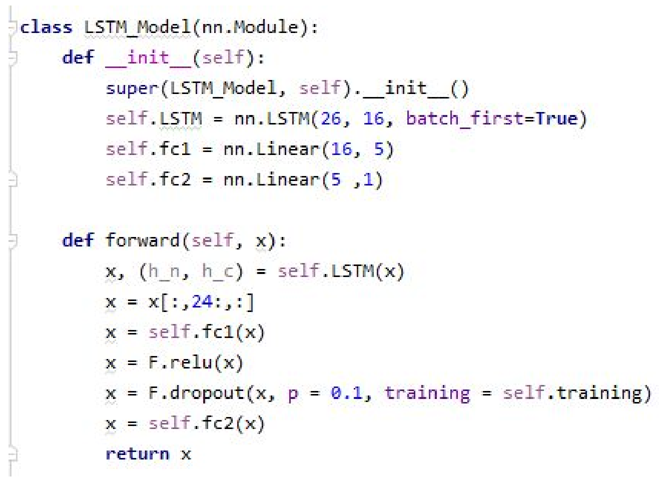


Figure 22. LSTM model illustration

The output of the LSTM layer will be fed into a 2 level dense layer to generate the final prediction value. The input data dimension is still 26, and we set the hidden layer of LSTM to contain 16 neurons. The two fully connected layers will further reduce the dimension. The output dimension of the first dense layer is 5 and the second is 1. The activation function is ReLU which is the same as the MLP model. The dropout layer is also used for regularization.



```
class LSTM_Model(nn.Module):
    def __init__(self):
        super(LSTM_Model, self).__init__()
        self.LSTM = nn.LSTM(26, 16, batch_first=True)
        self.fc1 = nn.Linear(16, 5)
        self.fc2 = nn.Linear(5 ,1)

    def forward(self, x):
        x, (h_n, h_c) = self.LSTM(x)
        x = x[:,24:,:]
        x = self.fc1(x)
        x = F.relu(x)
        x = F.dropout(x, p = 0.1, training = self.training)
        x = self.fc2(x)
        return x
```

Figure 23. LSTM model code

To train this model, we split data into units by every 48 hours. Then we shuffled the units and randomly selected 20% of training data out as a validation dataset. Here are the other hyper-parameters used for training:

Optimizer	Learning rate	Loss function	Batch size
Adam	1e-3	MSE	30

The only difference compared with MLP is the batch size is smaller due to LSTM using 48 hours of data as one unit.
We trained this model for 80 epochs and stored the validation result.

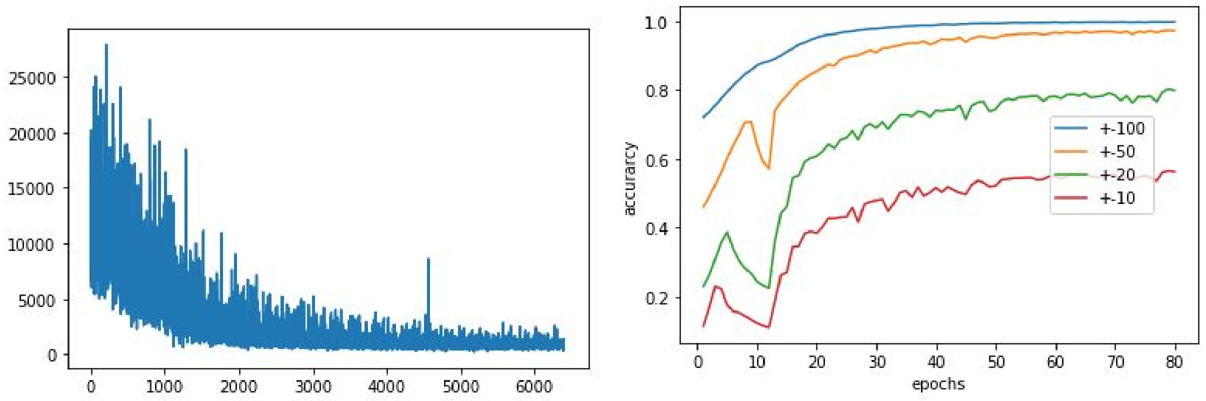


Figure 24. LSTM training loss by batch (left) and validation accuracy (right)

The model is also converged well based on the loss value curve, and the overall accuracy of validation data keeps increasing. So there is no sign of overfitting this time.
We notice there is a drop in the accuracy graph between epoch 5 to epoch 15. Even if we changed the hyper-parameters like the neuron numbers, the drop always exists. We suspect it may be because the model is learning from the large difference first which is dominating the loss values, such as the difference is more than one hundred. It will reduce the overall loss value, but the update for gradient may not be reflected in small tolerance criteria. From the graph, we can see the drop is significant when the tolerance is small such as ±10 and ±20. But for ±100, the curve is always growing.

Here is the final prediction performance on test data:

MAE	RMSE	MSE	±10 Acc	±20 Acc	±50 Acc	±100 Acc
15.45	24.90	670.17	56.20%	76.10%	94.20%	98.90%

Table 25. LSTM test result

Based on the result, there is a significant improvement compared with the MLP model. All of the three loss functions values have dropped. Moreover, there is a 13.3% increase for ±10 criteria, around a 10% increase for ±20, and a 4% increase for ±50. The LSTM model performs much better on small tolerance criteria and the prediction value is closer to the ground truth value.

Time Series Approach

Since our dataset was collected continuously by the hour, we also tried to predict the target value using a different representation of input data to fully utilize the time-series property of the dataset. We tried with 2 different approaches, each with unique data input.

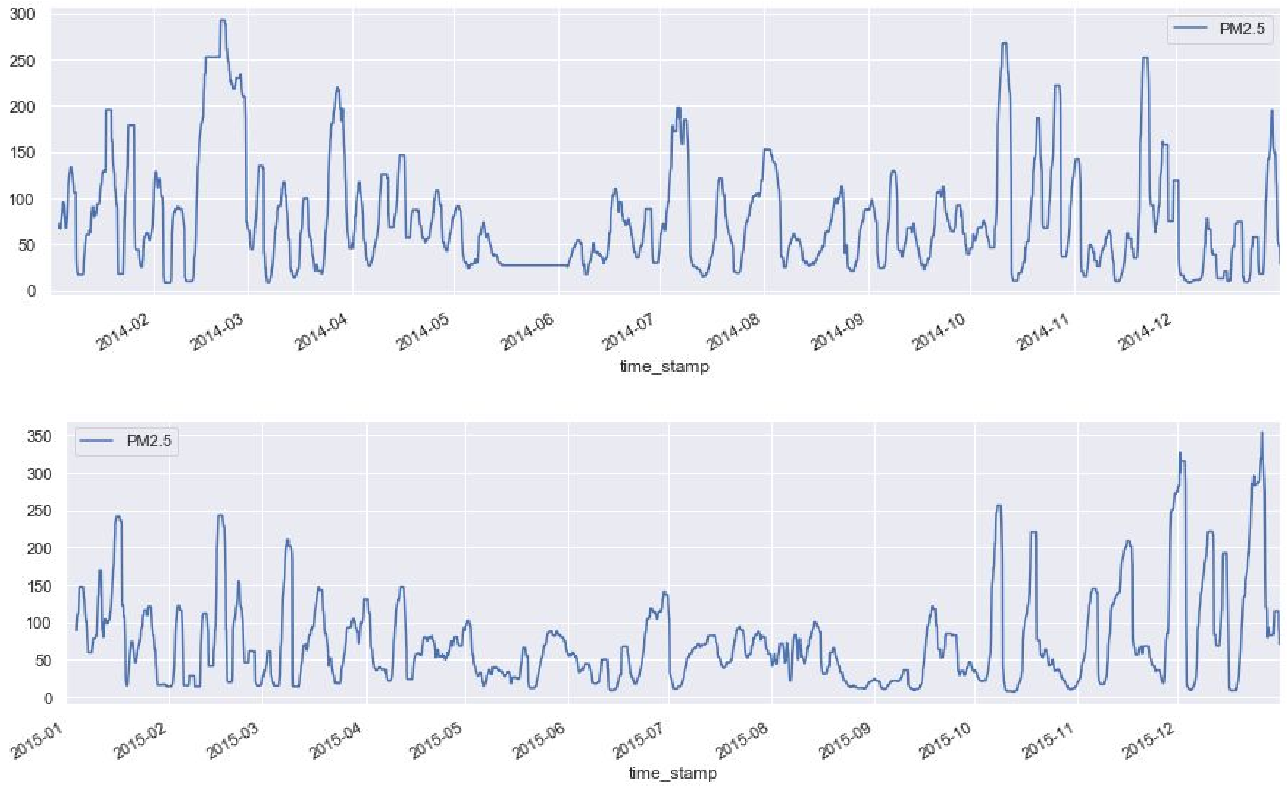


Figure 26. 3 Day Median Smoothing on PM2.5 for 2014 and 2015

Linear Regression With One Lag Variable

Lags are very useful in time series analysis because of a phenomenon called autocorrelation, which is a tendency for the values within a time series to be correlated with previous copies of itself. To make use of this property, we added one more column PM2.5_PREV containing previous hour PM2.5 data.

Because we have removed entries with any missing data from the dataset, our data is no more continuous. In such a case, neither the entry with missing data or the following entry must be discarded. We applied the same appending and removing rule to train, validation, and test data.

Results from the Linear Regression is below:

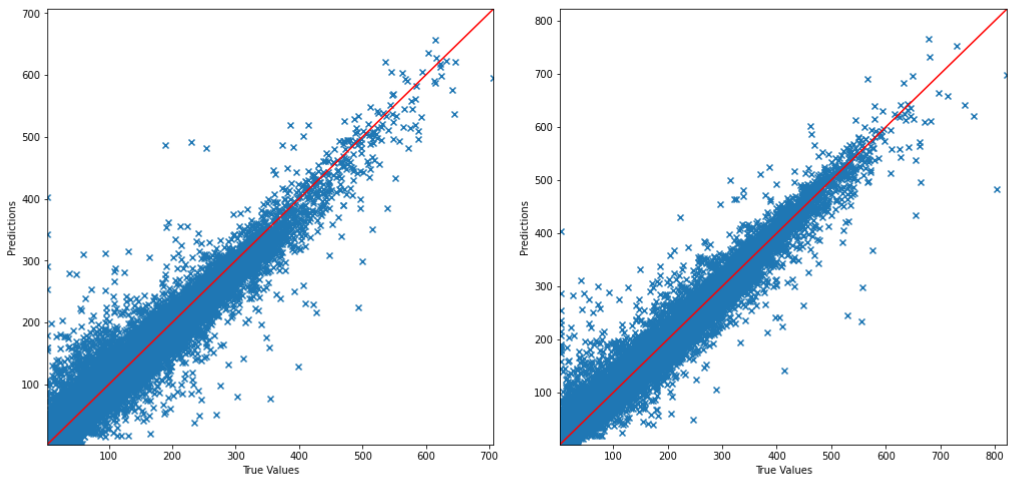


Figure 27. True PM2.5 vs Predicted PM2.5 on ValidationSet(left) and TestingSet(right)

We noticed this method is performing very well and it outperforms any other methods we explore above. However, we do think it is not fair to compare this model with others due to the fact that we used different input data. To justify whether this method is an improvement of a baseline prediction, which always predicts the current hour data with last hour data without considering any other parameters. A.k.a, we set the corresponding linear regression weights of PM2.5_PREV to 1 and all other weights to 0. Following is the comparison between the linear regression with one lag variable and baseline.

Model Name	MAE	RMSE	MSE	±10 Acc	±20 Acc	±50 Acc	±100 Acc
Linear Regression (with previous hour target data)	9.3	16.17	261.74	71.79%	90.14%	98.33%	99.72%
Baseline(predict with last hour data only)	10.66	20.97	439.86	68.76%	86.53%	96.82%	99.29%
Improvement	12.76%	22.89%	40.49%	4.41%	4.17%	1.56%	0.43%

Table 28. 1-lag linear regression with a baseline comparison

From the result, we can see the baseline model also performs well. There's a significant improvement for the linear regression with one lag variable. The above case is for with 1 lag data input, similarly, we can do it for n-lag comparison. N-lag here means the appended data is PM2.5 value from an hour ago.

Model Name	MAE	RMSE	MSE	±10 Acc	±20 Acc	±50 Acc	±100 Acc
Linear Regression (lag=1)	9.3	16.17	261.74	71.79%	90.14%	98.33%	99.72%
Linear Regression (lag=2)	12.91	20.7	428.84	57.10%	82.48%	96.93%	99.50%
Linear Regression (lag=4)	15.79	23.55	554.95	46.55%	75.29%	95.52%	99.41%
Linear Regression (lag=12)	17.68	24.87	618.69	40.59%	68.62%	94.58%	99.60%
Linear Regression (lag=24)	17.56	24.27	589.38	40.93%	68.35%	94.70%	99.37%
Linear Regression (lag=48)	17.59	24.43	597.11	40.57%	68.93%	94.39%	99.72%
Linear Regression (no lag)	17.68	30.68	642.31	42.07%	69.26%	94.08%	99.58%

Table 29. N-lag Results with Different Parameters

As expected, the n-lag method converges to the linear regression model as n increases. This implies that the PM2.5 value is autocorrelated to its historical data, and the most recent data has the highest correlation.

The Prophet Forecasting Model

Prophet is a time series forecasting tool developed by Facebook and available in both Python and R. The model is trying to fit several linear and nonlinear function of time as components including trend, seasonality, holidays and error term:

$$y(t) = g(t) + s(t) + h(t) + \varepsilon(t)$$

- g(t) is a piecewise linear or non-linear logistic growth curve for modeling the trend. The logistic growth has carrying capacity which determines the maximum target value as in Figure 31.
- s(t) is periodic effects (e.g yearly/weekly/daily seasonality) modeled by Fourier series as stated in the formula below. N is the Fourier order which determines the high or low frequency of changes, P is the period like 365 for yearly data or 7 for weekly data.

$$s(t) = \sum_{n=1}^{N} \left(a_n \cos\left(\frac{2\pi nt}{P}\right) + b_n \sin\left(\frac{2\pi nt}{P}\right) \right)$$

- h(t): user-specified holidays that allow inputs of customized irregular events.
- ε(t): error term for any unusual changes

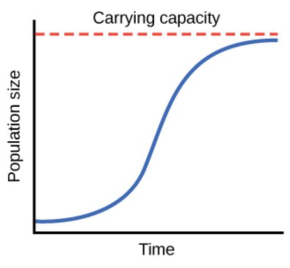
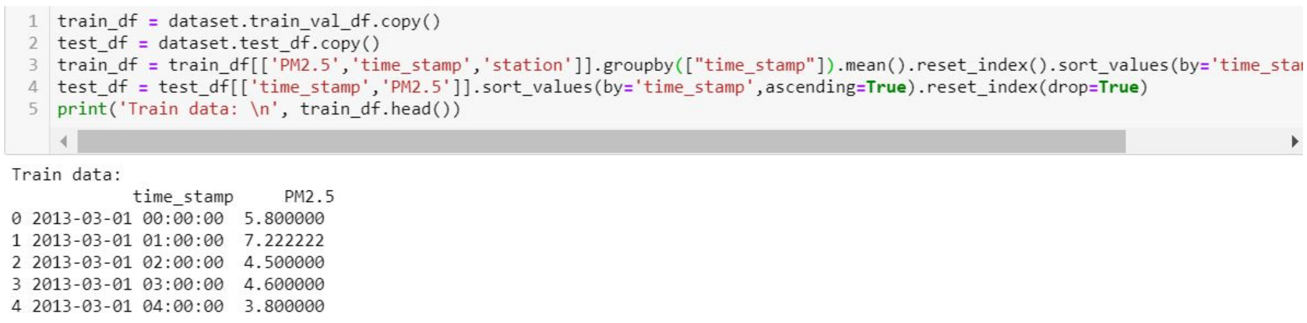


Figure 30. Logistic growth curve

Compared to classic time-series forecasting tools like ARIMA which need stationary data through differencing, Prophet provides completely automated prediction which is fast, productive and requires less effort in building a model. The training data only includes "time_stamp" in the format of DateTime and "PM2.5" which is the mean of the PM2.5 value of all stations at that time.

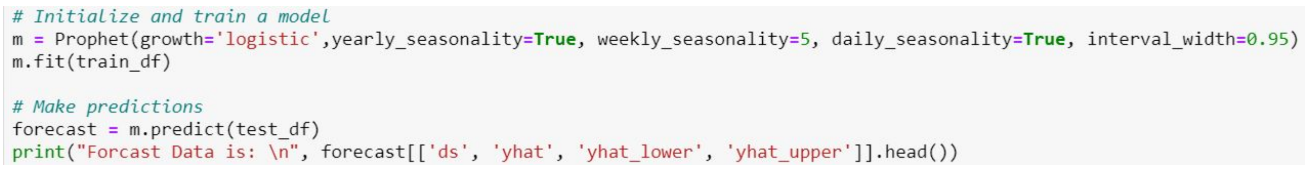


```
train_df = dataset.train_val_df.copy()
test_df = dataset.test_df.copy()
train_df = train_df[['PM2.5','time_stamp','station']].groupby(["time_stamp"]).mean().reset_index().sort_values(by='time_sta
test_df = test_df[['time_stamp','PM2.5']].sort_values(by='time_stamp',ascending=True).reset_index(drop=True)
print('Train data: \n', train_df.head())
```

```
Train data:
             time_stamp     PM2.5
0 2013-03-01 00:00:00  5.800000
1 2013-03-01 01:00:00  7.222222
2 2013-03-01 02:00:00  4.500000
3 2013-03-01 03:00:00  4.600000
4 2013-03-01 04:00:00  3.800000
```

Figure 31. Input data of Prophet

For the model setting, we tried tuning some hyperparameters to get better performance including change growth to be either linear or logistic and yearly/weekly/daily seasonality equals to 10 or 5. 'interval_width=0.95' sets the uncertainty interval to produce a confidence interval yhat_lower and yhat_upper around the forecast y_hat. Results show the performance variation is small with parameter tuning.



```
# Initialize and train a model
m = Prophet(growth='logistic',yearly_seasonality=True, weekly_seasonality=5, daily_seasonality=True, interval_width=0.95)
m.fit(train_df)

# Make predictions
forecast = m.predict(test_df)
print("Forcast Data is: \n", forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].head())
```

Figure 32. Parameters of Prophet

Prophet Model	**MAE**	**RMSE**	**MSE**	**±10 Acc**	**±20 Acc**	**±50 Acc**	**±100 Acc**
Prophet(linear growth, all N=10)	61.2	95.2	9029.0	11.80%	24.50%	63.90%	83.90%
Prophet(linear growth, week N=5)	61.3	93.9	8823.0	11.40%	23.80%	62.60%	83.60%
Prophet(linear growth, all N=5)	61.9	97.9	9485.6	12.20%	26.50%	65.20%	82.10%
Prophet(logistic growth, week N=5)	62.4	89.5	8006.1	10.10%	21.00%	52.30%	86.30%
Prophet(logistic growth, all N=5)	62.0	91.0	8287.3	10.80%	23.06%	53.72%	85.02%

Table 33. Evaluation of Prophet Model with different parameters

The predicted model yearly/weekly/daily seasonality trends and comparison of actual data from the years 2013 to 2017 show in Figure 34. The predicted yearly trend shows PM2.5 is lower in April to September and higher from October to March. This is explainable as the weather is cold from October to March and coal-fired heating is an important source for PM2.5 [3]. The predicted weekly trend shows lower PM2.5 on weekdays and higher on weekends, however, the actual weekly trend in 2014 is completely opposite to that in 2015, thus possibly the weekly trend is not identifiable. Lastly, for the daily trend, the predicted model shows PM2.5 from 4 am to 4 pm and higher at night which aligns with historical data in all years.

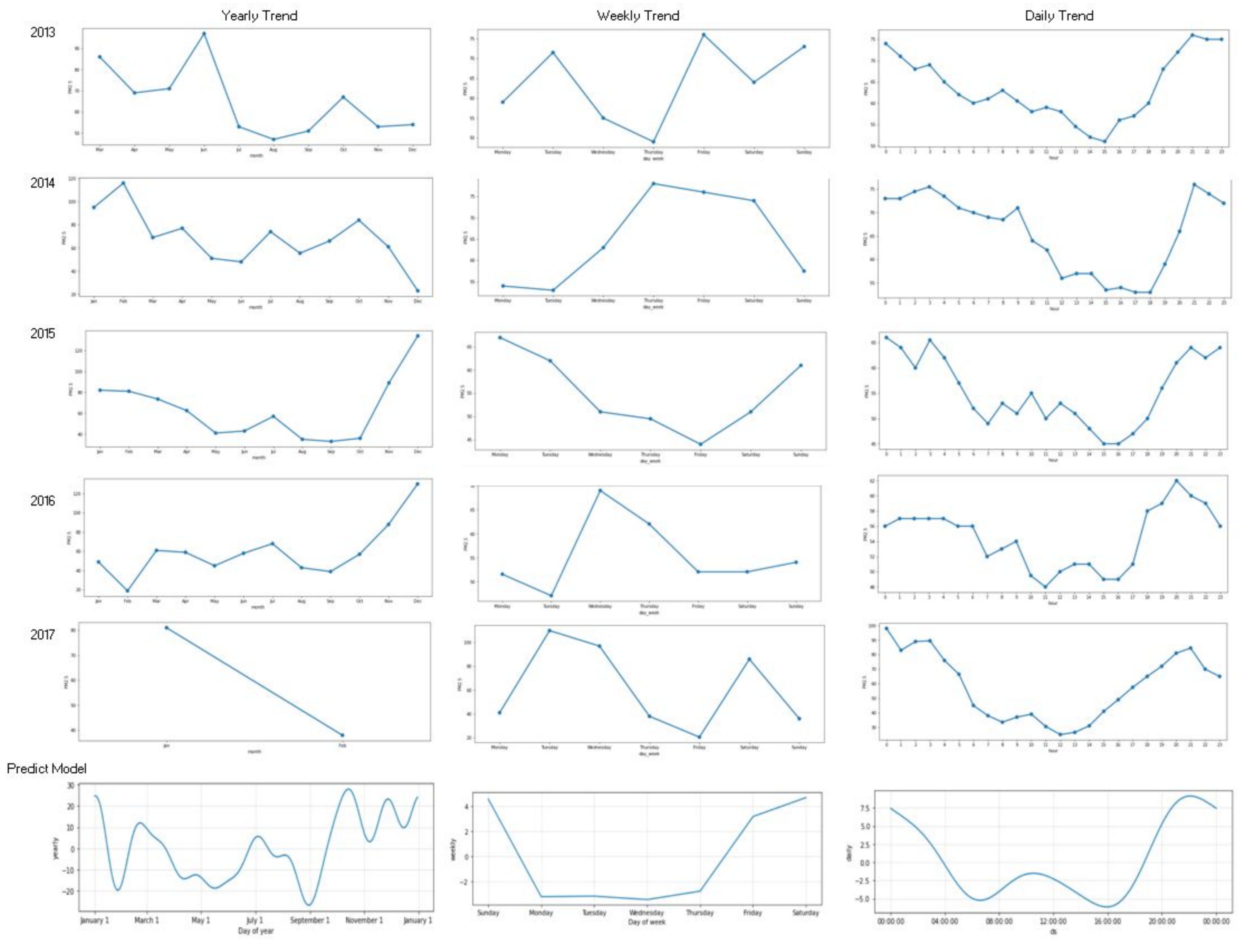


Figure 34. The year 2013-2017 Actual Data vs Predicted Model Trend

Summary and Discussion

We use Root Mean Squared Error(RMSE), Mean Absolute Error(MAE), and Mean Squared Error(MSE) to evaluate the accuracy of prediction models and make comparisons on the performance of different models. Below is a summary of all the models' prediction performance, highlighted the ones with the best performance:

Model Name	MAE	RMSE	MSE	±10 Acc	±20 Acc	±50 Acc	±100 Acc
Linear Regression	17.68	30.68	642.31	42.07%	69.26%	94.08%	99.58%
LARS Lasso Regression	17.57	26.06	679.34	45.82%	70.20%	93.52%	99.43%
Linear Support Vector Regression	16.59	24.76	613.14	48.88%	71.63%	94.16%	99.59%
MLP NN	20.64	31.49	999.88	42.90%	65.50%	90.30%	98.40%
Decision Tree	18.47	31.71	1005.78	50.70%	69.49%	90.65%	98.29%
XG Boost	14.66	23.63	558.75	52.43%	75.21%	96.68%	99.35%
LSTM RNN	15.45	24.90	670.17	56.20%	76.10%	94.20%	98.90%
Prophet	62.4	89.5	8006.1	10.10%	21.00%	52.30%	86.30%

Figure 35. All models test result

In conclusion, most of the models perform well and reach above 90% accuracy with high tolerance criteria. The XGBoost model achieves the lowest error and the LSTM RNN model has the best accuracy for ±10 and ±20 tolerance ranges.

Based on the linear regression result, PM10 is the main indicator when predicting PM2.5, other important features like CO, DEWP, and WD_NW also have a strong correlation to PM2.5. DEWP indicates the moisture amount in the air. The higher DEWP value the higher moisture content at a specific temperature. The moisturized air can hold particulars. Below a certain point, DEWP is a positive correlation with PM2.5. The coefficient of determination in the training process can achieve 0.928, which indicates the variance in the dataset can be well-explained by those input variables.

For the neural network approach, the LSTM model performance dominates the MLP model because of the shared hidden features learned from historical data and the gating concept. Another finding is that adding more neurons during the fine-tuning process will not help to enhance the model accuracy. It may be due to the data size not being big enough for feeding a too complex model.

We also tried the time series prediction model. In the yearly trend, we find PM2.5 drops in April and increases in October when coal-fired heating contributes a lot to PM2.5. Besides, the daily trend shows PM2.5 is relatively higher at night and lower at noon, which contradicts “common sense” that people choose to workout in the morning to avoid the worst pollution. The weekly trend varies a lot in different years, which may be one reason why the Prophet model does not perform well for this project and we can improve it with more knowledge of the change points in different years.

Further steps still can be taken based on our work. For data preprocessing, we didn’t consider outlier removal. Since we used min-max normalization to scale our dataset, which is very sensitive to the presence of outliers. This could possibly increase the error. A good direction to focus on is quantile transformer, which is a robust normalization method that maps all the data to range 0 and 1 and it transforms outliner datapoint and non-outlier data point well. We didn’t try this because of the project timeline and it requires a reevaluation on all the models.

Reference

[1] "Beijing Multi-Site Air-Quality Data Set | Kaggle." 8 Nov. 2019, https://www.kaggle.com/sid321axn/beijing-multisite-airquality-data-set. Accessed 26 Apr. 2020.
[2] V.Morde, V.A.Setty (2019, April 8). XGBoost Algorithm. Retrieved from https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d
[3] Beijing PM2.5 source analysis Retrieved from http://www.cnki.com.cn/Article/CJFD2005-HJKX200505000.htm
[4] Prophet API. Retrieved from https://facebook.github.io/prophet/docs/quick_start.html
[5] S.Ahmed (2018, September 19). An Intuitive Perspective to Linear Regression. Retrieved from https://hackernoon.com/an-intuitive-perspective-to-linear-regression-7dc566b2c14c

Tools

Library used: sklearn, xgboost, PyTorch, fbprophet

Appendix

Detailed Output from Program (GitHub Webpage):

Preliminary analysis

Basic Models (Linear Regression, SVM, Decision Tree, XGBoost)

N-lag Linear Regression

Neural Networks - MLP

Neural Networks - RNN(LSTM)

Prophet