# Project 5 Pollution Vision

*This manuscript (permalink) was automatically generated from Saran-Wang/dsproject@ac40f98 on December 6, 2020.*

## Authors


- **Shiyuan Wang**
  · Saran-Wang
  Department of Civil and Environmental, University of Illinois

- **Weiqi Ni**
  · weiqini
  Department of Civil and Environmental, University of Illinois; Department of Environmental and Resources, Zhejiang University

- **Gemma Clark**
  · 441gclark
  Department of Civil and Environmental, University of Illinois

- **Xueao Li**
  · XueaoLi
  Department of Civil and Environmental, University of Illinois

## Abstract

# Introduction

## Literature Review

There are many studies using digital camera and advanced algorithm to estimate the concentrations of Particulate Matters. Hong et al. [1] developed a novel method of predicting the concentrations and diameters of outdoor ultrafine particles using street-level images and audio data in Montreal, Canada. Convolutional neural networks, multivariable linear regression and genralized additive models were used to make the predictions. Wong et al. 2007 present an image processing method for estimating concentrations of coarse particles (PM10) in real time using pixels acquired by an internet video surveillance camera. In this paper, the authors present formulas for predicting particulate matter based on optical physics including light absorption, scattering, and reflection. They do not use machine learning tactics to estimate pollution concentrations, but their model results in root mean square error values of around 4 μg/m3.

# Exploratory Data Anlysis

1. Variables Explanation

**Table 1:** Variables Explanation

| Data Fields | Explanation |
|---|---|
| Temp(C) | ambient temperature |
| Pressure(kPa) | air pressure |
| Rel. Humidity | relative humidity |
| Errors | if the air measurement equipment has error during sampling (0=no) |
| Alarm Triggered | if any instrumental warning shows during sampling (0=no) |
| Dilution Factor | an instrumental parameter (should close to 1) |
| Dead Time | another instrumental parameter (ideally close to 0) |
| Median, Mean, Geo. Mean, Mode, and Geo. St. Dev. | parameters describe particle sizes, which can be ignored |
| Total Conc. | an output variable from the instrument that should not be used |
| image_file | the visual information of the traffic condition, corresponding to an image in the “frames” directory |
| Wind_Speed | the wind velocity during sampling |
| Distance_to_Road | the distance between camera and road |
| Camera_Angle | the angle of incidence between the camera and the road |
| Elevation | the elevation between the camera and the breathing zone |
| Total | the total measured particle number concentration (# / cm3) This is the dependent variable |

2. Data Cleaning

- Delete the useless columns in the dataset
- Delete the rows with equipment error during sampling

3. Visualization of the distributions of varibales

Figure 1 shows that “Wind_Speed”, “Camera_Angle”, “Distance_to_Road” and “Elevation” are all in discrete distributions, while “Temp(C)” are in continuous distribution. “Pressure(kPa)” has four clusters. It should also be noted that the “Dead Time” almost shares the same distribution as “Total”.

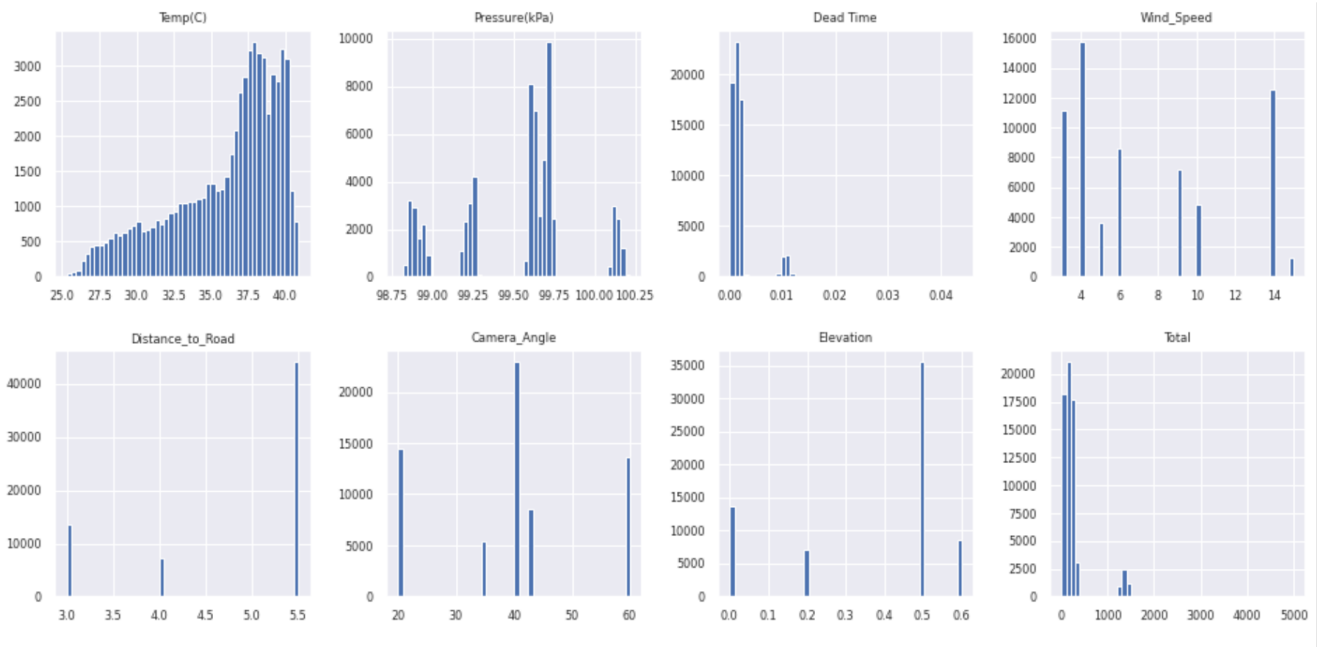


**Figure 1: Variables Distribution**

4. Correlations among variables

From the correlation map 2 we could see that “Dead Time” are extremely correlated with “Total”, with a coefficient of 1, followed by “Camera_Angle”, “Pressure(kPa)” and “Distance_to_Road”, with coefficient of 0.52, 0.49, 0.44 respectively. Here you may be curious why “Dead Time” could be so closely related to “Total”, and there is one possible explanation: Actually, “Dead Time” is an instrument parameter, and if there are more PM concentrations in the air, the instrument need more time to process, and vice versa.

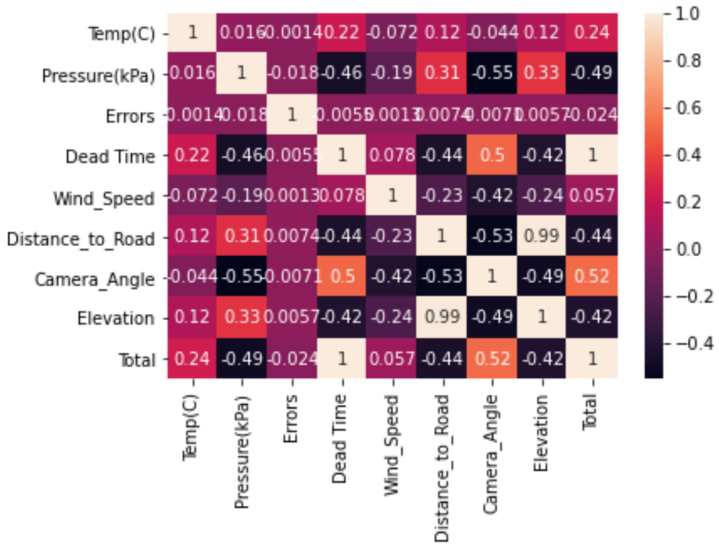


**Figure 2: Variables Correlations**

# Model

## Shiyuan's Model

My model setup splits into two part, the first is image data extraction, the second is the selection of appropriate model to fit this dataset.

## Image Extraction

First I want to digitize images by extracting image features, there are mainly 6 features I want to extract: RGB, image luminance, image contrast, image entropy, transmission and amount of haze removed and number of cars on streets.

1. RGB

The RGB color model is one of the most straightforward parameters describing an image. Intuitively, in this case, we may expect more blueness and greenness if the PM concentrations are low since the color of tree and sky would be brighter when the air conditions are good. For each image, after deriving the RGB of each pixel, we take the average of them, and then divide each value by 255 to normalize it. The figure below 3 shows the distributions of RGB in this dataset. We can see that they are nearly normally distributed with mean 0.45, 0.55 and 0.35 respectively. For blueness, we could see a second peak at around 0.42.

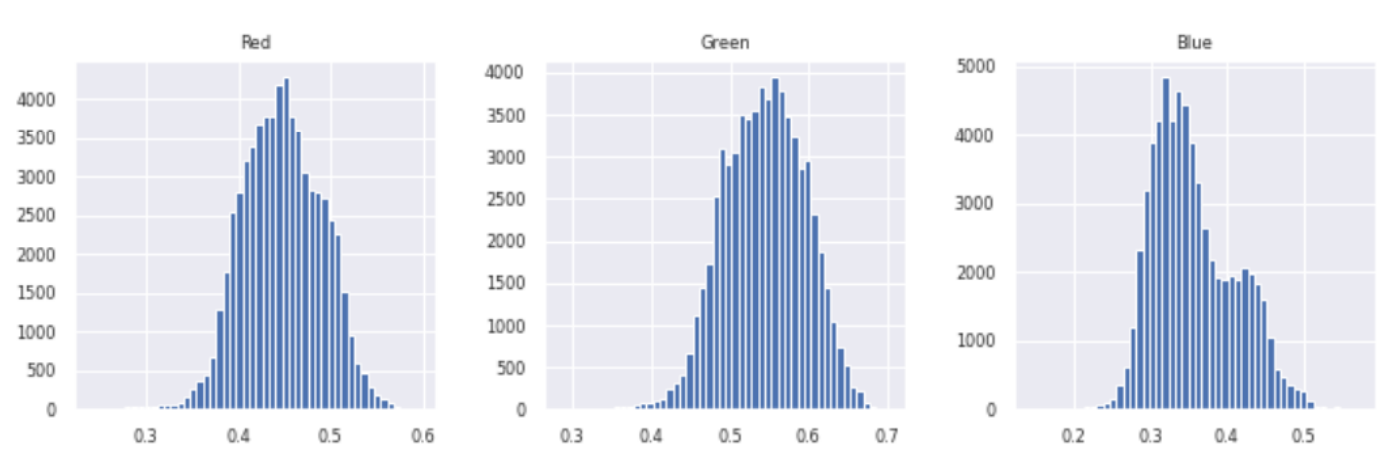


**Figure 3: RGB Distribution**

2. Luminance

Like RGB, luminance is also a very basic parameter describing an image, which could be an indicator of how bright the image will appear. The luminance of each image is calculated by taking the average of the luminance intensity of each pixel. From figure 4 we could also see that it's also normally distributed with a mean of around 130.

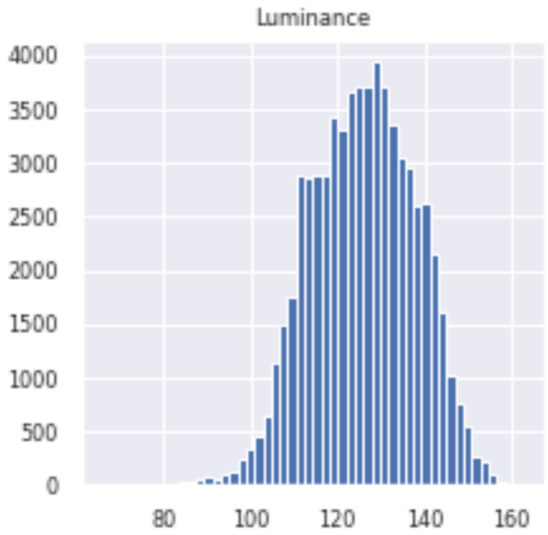


**Figure 4: Luminance Distribution**

3. Contrast

The image contrast is defined as the difference between the max and min luminance intensity of an image. Study [2] shows that the higher the PM concentrations, the lower contrast would be. It makes sense since the image would become vague and lighter when there are more particulate matters in the air. And often, one image would have pixels with the highest intensity of 255, as well as the lowest intensity of 0. Therefore, we can't see much difference if we want to derive the absolute contrast, since it would be 1 for most of those images. Therefore, we use root mean square of image intensity to describe image contrast.

$$Absolute_Contrast = \frac{I(i_{max}, j_{max}) - I(i_{min}, j_{min})}{I(i_{max}, j_{max}) + I(i_{min}, j_{min})} \quad (1)$$

$$RMS_Contrast = \sqrt{\frac{1}{MN}\sum_{i=1}^{N}\sum_{j=1}^{M}(I(i,j) - avg(I))^2} \quad (2)$$

where I(i,j) is luminance intensity at (i,j) pixel.

From figure 5 we could see that the distribution is a little bit right-skewed with a small peak at around 35, and a larger one at around 50.

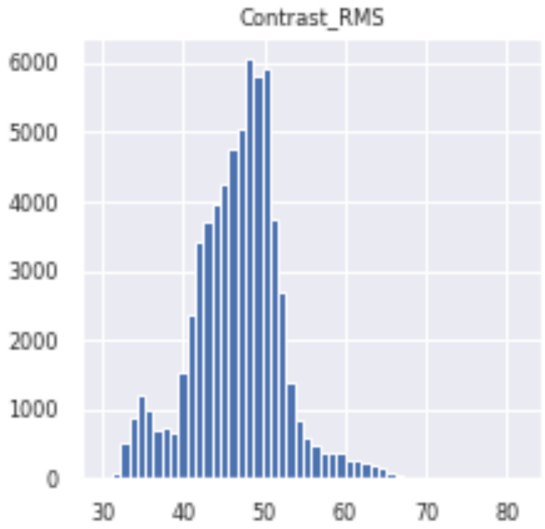


**Figure 5: Contrast Distribution**

4. Entropy

Image entropy is a statistical measure of randomness that quantifies information contained in an image. Usually, an image would lose its details with the increasing PM concentrations, and the image entropy will decrease as a result [2] . To do the calculation, I first converted the original RGB image to grayscale image and then used a module within python: *skimage* to calculate image entropy directly. The example code is shown as below:

```
colorIm =
        Image.open('../input/pollutionvision/frames/frames/video06082020_0.jpg

greyIm = colorIm.convert('L')
ImContrast = skimage.measure.shannon_entropy(greyIm)
```

Figure 6 is the original figure and figure 7 shows its entropy.


**Figure 6: Original Image**

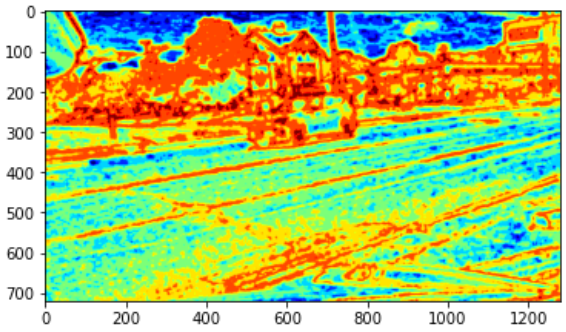


**Figure 7: Entropy**

5. Transmission and amount of haze removed

6. Number of cars on streets

Also, I take the number of cars on streets into account. Intuitively, the more cars on the street, the higher PM concentrations would be. There is a very efficient library in python called *cvlib*, within which a function called *object_detection* could detect the number of different objects appearing on an image. The example code is shown as below:

```
im =
        cv2.imread('../input/pollutionvision/frames/frames/video06082020_0.jpg
bbox, label, conf = cv.detect_common_objects(im)
output_image = draw_bbox(im, bbox, label, conf)
number_of_car = label.count('car')
```

As we can see in figure 8, the left image has 2 cars on street, and we can detect exactly two cars; while figure 9 has no car on street but 5 cars parking in the parking lot, and we could detect 5 cars.

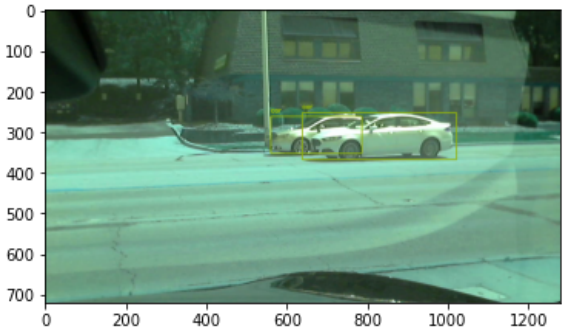


**Figure 8: Two cars detected**

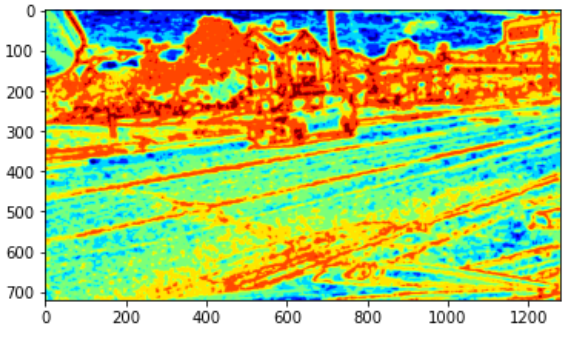

**Figure 9: Five cars detected**

Here comes the problem, this function could only detect the number of cars appearing on an image but can't identify which is in motion. But the moving cars are actually the ones which contribute to PM concentrations at the very moment. However, in this case, I just keep the original detection results, since if there are more cars in the parking lot, I just assume it's a traffic busy day, on which the PM concentrations would be higher than normal days.

7. Correlations among variables

Here we plot out the spearman correlations among those features in figure 10, the last column shows the spearman correlations between each feature and the Total PM concentrations. As we can see, the dead time, which is an instrument parameter, is closely correlated with PM concentrations, followed by Pressure, RGB, luminance and temperature. However, since the dataset is rather complicated, the correlations may mean nothing. Actually, the different combination of different features may have various impacts on the results of our model. And the correlations just provide us with a straightforward perception.

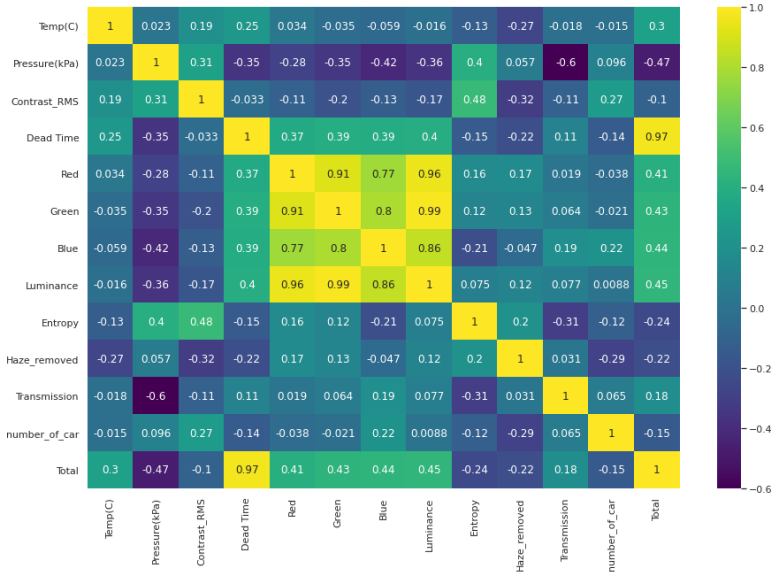


**Figure 10: Variables Correlations (include digitized image data)**

## Model Selection

As mentioned before, we can't select the features barely based on their correlations with PM concentrations, since I have both numerical data and digitized image data, which could be very complicated. Therefore, I selected different combinations of features and run the model several times to select the one with best performance. At first, I tried the Neural network, but it doesn't do well in this dataset, with an MSE of around 800. Then I calculated the model accuracy using *cross_val_score*, and then I switch to random forest, which gives me an accuracy of 0.997. With the help of * GridSearchCV*, I could decide on the parameters for random forest: `RandomForestRegressor(max_depth=20, n_estimators=1000, random_state=3)`. With those parameters, I could get an RSME around 11.

## Gemma's Model

I used three different approaches to predict pollution concentrations: building a neural network using the numeric data, creating a neural network using the image data, and developing a random forest model using the numeric data.

For all three models, I performed the same first initial steps. I read in the "csv" files containing the training dataset and test dataset and removed the variables that should not be included in the model. These variables included parameters related to the particulate size and shape, superfluous instrument parameters, and variables with a standard deviation of 0. For each of the 64961 data points, there was one image and eight numeric variables: temperature, pressure, errors, dead time, wind speed, distance to road, camera angle, and elevation. There were no missing values in the dataset. I then randomly split the training dataset into a validation dataset (20% of the original training dataset) and a new training dataset (80% of the original training dataset).

### Neural Network (Numeric Data)

To prepare the numeric data for the neural network model, I set the eight numeric variables to be the independent "x" variables and the total pollution to be the dependent "y" variable in a tensorflow dataset. I set the batch size to be 50 and created a "Sequential" keras model. My model had four layers: two "relu" layers with 30 units each, a "sigmoid" layer with 30 units, and a linear layer with 15 units. Using a learning rate of 0.0005, loss of mean square error, and 30 epochs, I compiled the model and tested it on the validation dataset. The mean square error converged at around 1,500-1,800 depending on the random training and validation dataset generated in the initial setup.

### Convolutional Neural Network (Image Data)

To prepare the image data for the neural network, I created a function that loaded the image from the image name, randomly flipped it vertically, randomly flipped it horizontally, and then randomly cropped the image to be 72 x 128 (from the original size of 720 x 1280). I initially tried using the entire image, but when compiling the model, my computer ran out of memory. Using a batch size of 25 and "imagenet" weights, I created a model using "applications.Xception" and added a normalization layer that normalized the image data from (0, 255) to (-1, +1). The weights of the normalization layer were the mean (0 + 255)/2 = 127.5 and variance (in this case set to be the square of the mean). My model used GlobalAveragePooling2D, had a Dropout at 0.5, and activation of "softmax." Using a learning rate of 0.00005, "optimizers.Adam," loss of mean squared error, and 10 epochs, I fit the model to my validation dataset and regularly had mean square error values exceeding 170,000 for each of the different randomly selected validation and training datasets created in the initial setup.

### Random Forest (Numeric Data)

For the random forest model, I set the eight numeric variables from the training dataset to be the independent "x" variables and the corresponding total pollution from the training dataset to be the dependent "y" variable. I also separated the validation dataset into the independent variables and dependent variable. My random forest model had 1000 "n_estimators" (trees), used mean square error as its criterion, had a maximum depth of 20, and had a minimum sample split of 2. I fit the random forest model using the training dataset and then used the model to make predictions for the validation dataset. The resulting mean square error was around 140, which was much lower than the mean square error produced by the neural networks for the numeric and image data. Looking at the features that had the most influence on the random forest model, the dead time had 100-1000-fold more impact on the predicted pollution than any of the other independent variables. This makes

sense because I now know that the dead time is an instrument parameter stating how long the instrument has to “think” to measure the pollution, so longer dead times would be associated with higher pollution.

Since my random forest model performed best on my validation dataset, I used the entire original training dataset (not split into training and validation data) to create a random forest model with the same parameters. I then used this model to predict pollution values in the test dataset which resulted in a final mean square error of 124 or root mean square error of 11.2.

## Weiqi’s Model

# Xueao’s Model

## Conclusion

All of the team members found random forest models to produce the best results with the lowest root mean square error. While each member used different parameters for her model, the final predictions had root mean square error values of less than 20. Based on our results, we conclude that machine learning can be used to approximate particulate matter with the variables we had available, but a better model will be needed to produce more accurate predictions.

# References


1. **Predicting outdoor ultrafine particle number concentrations, particle size, and noise using street-level images and audio data**
   Kris Y. Hong, Pedro O. Pinheiro, Scott Weichenthal
   *Environment International* (2020-11) https://doi.org/ghnh6n
   DOI: 10.1016/j.envint.2020.106044 · PMID: 32805577

2. **Particle Pollution Estimation Based on Image Analysis**
   Chenbin Liu, Francis Tsow, Yi Zou, Nongjian Tao
   *PLOS ONE* (2016-02-01) https://doi.org/ghnjkc
   DOI: 10.1371/journal.pone.0145955 · PMID: 26828757 · PMCID: PMC4734658