4-6-2023

A Graphical User Interface using Spatiotemporal Interpolation to determine Fine Particulate Matter Values in the United States

Kelly M. Entrekin
Georgia Southern University

A Graphical User Interface using Spatiotemporal Interpolation to determine Fine Particulate Matter Values in the United States

An Honors Thesis submitted in partial fulfillment of the requirements for Honors in *Computer Science*

By
Kelly Entrekin

Under the mentorship of *Dr. Lixin Li*

ABSTRACT

Fine particulate matter or $PM_{2.5}$ can be described as a pollution particle that has a diameter of 2.5 micrometers or smaller. These pollution particle values are measured by monitoring sites installed across the United States throughout the year. While these values are helpful, a lot of areas are not accounted for as scientists are not able to measure all of the United States. Some of these unmeasured regions could be reaching high $PM_{2.5}$ values over time without being aware of it. These high values can be dangerous by causing or worsening health conditions, such as cardiovascular and lung diseases. Within this study, fine particulate matter values were interpolated at centroids of all counties in the United States throughout 2009 using the Python programming language and a spatiotemporal Inverse Distance Weighting (IDW) interpolation method. Machine learning concepts, such as ten-fold cross-validation, and error statistics were used to assess the accuracy of the estimated $PM_{2.5}$ values. The created Python programs display a graphical user interface for easy interaction between a user and the system. This allows the system to be used by more than just experts. The values reported in this study can also be used to determine if unmeasured county areas are reaching unsafe $PM_{2.5}$ values throughout the year.

Thesis Mentor: Dr. Lixin Li

Honors Dean: Dr. Steven Engel

April 2023

Department of Computer Science

Honors College

Georgia Southern University

Acknowledgements

I would like to thank Dr. Lixin Li for all her support throughout the completion of this research and thesis. Her mentorship capabilities encouraged me to push farther and learn more independently. She encouraged my growth and guided me alongside professional development. She has inspired me to continue exploring my passions inside the field of computer science. This research would not have been possible without her and the resources she provided.

1 Introduction

Fine particulate matter or $PM_{2.5}$ can be described as solid and liquid particle pollution that has a diameter of 2.5 micrometers or smaller [1]. These pollution particles can be formed from basic human creations and interactions, such as gasoline, oil, dust, smoke, and dirt [2]. Some common examples that emit $PM_{2.5}$ values include cooking, driving a car, and operating a fireplace. The United States Environmental Protection Agency measures $PM_{2.5}$ values at regular monitoring sites, or certain counties, in the United States throughout the year. The agency measures these points in micrograms and has standards for how high these levels should be before they are dangerous.

A higher exposure to this particulate matter can be dangerous as it can cause or worsen long term health concerns. These particles are able to travel deeper into the lungs and bloodstream because of how tiny they are. Exposure to these particles could result in cardiovascular and respiratory effects, such as heart attacks and bronchitis. Those more susceptible to health issues from particulate matter include children, older individuals, and those with pre-existing conditions; for example, asthma or abnormal heart rhythms. It is even suggested that higher exposure to pollution particles can lead to premature death [3]. Since pollution particles are created from everyday human activities and could have serious long-term health effects, $PM_{2.5}$ values become crucial to study.

While the data the EPA collects is important, there are various areas that are not accounted for that could be reaching high risk levels, not allowing residents to have the chance to be aware of what they could be exposed to. As these particles can be so damaging, the calculation for unmeasured centroids of all counties throughout the United

States becomes an important issue to address. This research focuses on determining $PM_{2.5}$ values for unknown locations in the United States.

Spatiotemporal interpolation can be helpful in finding $PM_{2.5}$ values that are not measured on a daily basis. Spatiotemporal interpolation occurs when a set of known data points are used to estimate a value in space and time of an unknown data point. Specifically, an extension-based method of interpolation was used for this research instead of a reduction-based method. This means that time was treated as another dimension in space instead of being reduced to regular spatial interpolation, which only uses space [4]. Time was used as its own variable throughout the interpolation formulas and calculations.

There are several types of interpolation, but this research focuses on inverse distance weighting interpolation with the extension approach. Inverse distance weighting (IDW) expresses that values that are closer to the point of interest are more related to that point than those that are farther away [5]. IDW with the extension approach eliminates the error of picking points that are farther away from the point of interest. Inverse distance weighting is calculated from the Euclidean distance, which is the distance between two points in a coordinate plane. Those with the shortest distance to the point of interest are called the nearest neighbors. The nearest neighbors are used against each other to find the weight each of them have on the overall $PM_{2.5}$ value of an unknown centroid point. More information about specific types of interpolation and how they relate to IDW can be found in [4]. Along with this, research has already been completed to interpolate the same county $PM_{2.5}$ values using a different type of interpolation called shape function interpolation [6]. The shape function interpolation method is compared

with IDW interpolation in [1] using different time scales. It was observed in that study that shape function interpolation is more efficient, but harder to implement than IDW. This study will test the accuracy of implementing IDW.

The accuracy of the spatiotemporal interpolation methods can be calculated using k-fold cross validation and error statistics. More specifically, 10-fold cross validation is used. This means that ten percent of the known values were removed from a file and used as unknown points. The other ninety percent of the file was used to determine the fine particulate matter value for each point in the ten percent [7]. Then, the actual and estimated fine particulate matter values were compared using estimated error values to determine how accurate the methods were. These statistical error values include the mean absolute error, the mean squared error, the root mean squared error, and the mean absolute squared error.

With this background knowledge, spatiotemporal inverse distance weighting interpolation with the extension approach is used to determine the fine particulate matter value of unmeasured centroid locations throughout the United States in 2009.

2 Objective and Method

2.1 Inverse Distance Weighting using Spatial Interpolation

Inverse distance weighting (IDW) based interpolation can be used in a 2D space for regular spatial interpolation [1, 8]. The following formulas are used for this type of interpolation:

$$d_i = \sqrt{(x_i - x)^2 + (y_i - y)^2}$$

$$\lambda_i = \frac{(\frac{1}{d_i})^p}{\sum_{k=1}^{N} (\frac{1}{d_k})^p}, \qquad w(x, y) = \sum_{i=1}^{N} \lambda_i w_i$$

The Euclidean distances are found using the d_i formula, which describes the distance between the points (x_i, y_i) and (x, y). These distances are used to find the weights each distance has compared to each other, which is described in the λ_i formula. Once the weight each distance has to the overall $PM_{2.5}$ value is found, they are multiplied by their known $PM_{2.5}$ value, w_i. Each distance weight that is multiplied by its original $PM_{2.5}$ value is added together to find the $PM_{2.5}$ value of the unknown point, which is defined in the w(x, y) formula.

2.2 Inverse Distance Weighting using the Extension-Based Spatiotemporal Interpolation

IDW can be used in a 3D space with the extension based approach for spatiotemporal interpolation, which treats time as another dimension in space [1]. This type of interpolation can be found using the following formulas:

$$d_i = \sqrt{(x_i - x)^2 + (y_i - y)^2 + c^2(t_i - t)^2}$$

$$\lambda_i = \frac{(\frac{1}{d_i})^p}{\sum_{k=1}^{N} (\frac{1}{d_k})^p}, \qquad w(x, y, ct) = \sum_{i=1}^{N} \lambda_i w_i$$

The Euclidean distances are found using the d_i formula in a 3D space, which describes the distance between the points (x_i, y_i, ct_i) and (x, y, ct). The concept of time is defined by c which is the spatial distance unit over the time unit. The Euclidean distances calculated are used the same way as the spatial IDW formulas; to find the weights and overall $PM_{2.5}$ value. The overall $PM_{2.5}$ value of the unknown point in space and time is defined in the w(x, y, ct) formula.

2.3 Time Dimension Choice for Extension-Based Spatiotemporal Interpolation

Four time scales were tested in prior research [1]. This research expands upon the time scale labeled A which uses the time dimension as c = 1. Each day of the year was given a value between 1 and 365 based on the day of the year it was. This means that January 1, 2009 was given the value 1 and December 31, 2009 was given the value 365. All months and days within the year were given their appropriate number for testing.

2.3 Cross Validation: k-fold

The data found for each $PM_{2.5}$ value was checked for accuracy using k-fold cross validation. Cross validation involves splitting the data into two groups; a training set and a validation set. For k-fold, the data is split into k equal folds. One of the k folds is used as the validation set, while the other k-1 folds are combined and used as the training set [7]. This form of validation is completed k times until all the k folds are used as the validation set. Specifically, 10-fold cross validation is used to provide an estimate of the accuracy of the $PM_{2.5}$ calculations at the selected time dimension.

10-fold cross validation means that the data was split into ten equal parts. Ten percent of the data is used as the validation set, while the other ninety percent is used as the training set. This is completed ten times to make sure that each equal part is used as the validation set and left out from the training set. The validation set $PM_{2.5}$ values are then compared with their original $PM_{2.5}$ using error statistics.

2.4 Error Statistics

High-level statistical error values were used to compare the values found using the 10-fold cross validation to see how accurate the created Python codes were using the time dimension selected. These error statistics include the Mean Absolute Error (MAE),

the Mean Square Error (MSE), the Mean Absolute Relative Error (MARE), and the Relative Mean Square Error (RMSE). All of these error statistics calculate the difference between the actual and predicted $PM_{2.5}$ values. The MAE calculates the average of the difference between the predicted and actual $PM_{2.5}$ values, while the MARE calculates how large of a difference there is between the MAE and the actual $PM_{2.5}$ values [9]. The MSE calculates the average of the squared difference between the measured and actual $PM_{2.5}$ values, while the RMSE calculates how large of a difference the MARE is from the expected values. This is also known as calculating the variance and the standard deviation [10]. The formulas are indicated below.:

$$MAE = \frac{\sum_{i=1}^{N} |I_i - O_i|}{N} \qquad MSE = \frac{\sum_{i=1}^{N} (I_i - O_i)^2}{N}$$

$$MARE = \frac{\sum_{i=1}^{N} \frac{|I_i - O_i|}{O_i}}{N} \qquad RMSE = \sqrt{\frac{\sum_{i=1}^{N} (I_i - O_i)^2}{N}}$$

For each of these formulas, the total number of calculated centroid points is defined by N. The interpolated value for each centroid is denoted by I_i. The actual measured value for each centroid is defined by O_i.

2.5 Python Programming Language

The Python programming language was implemented using PyCharm. Python was the chosen programming language as it contains several packages in its library for implementing a graphical user interface (GUI) easily. PySimpleGUI was used to create an interface with prompts, input text boxes, and buttons [11]. The first code that was created allows a user to input any file with known $PM_{2.5}$ values and ask for the $PM_{2.5}$ value of at any unknown x and y coordinate with any identification number. The user is able to input the power and nearest neighbors value they would like to use. The interface

then displays a screen with the calculated value and asks the user if they would like to find the $PM_{2.5}$ value at another unknown location. The accuracy of the first codes calculations were checked using hand calculations with the same known locations, unknown locations, power value, and nearest neighbors.

After the completion and accuracy of the first code, a second code could be created to allow a user to input a file of known $PM_{2.5}$ values, a file of unknown centroid locations and a filename for output. A power value of one and a nearest neighbor value of seven was used for each of the calculations. These values were chosen as stable values to also be used in the cross validation.

Another set of codes was created to start the 10-fold cross validation process of separating the $PM_{2.5}$ data values into 10 equal folds. One code in this set produced twenty files; ten files containing a random ten percent and another ten files containing the matching ninety percent. The ten percent files were then placed into another code that removed their original $PM_{2.5}$ value and placed it inside another file. Then, these files were used in the second code with the ten percent files as the file of unknown centroid locations and the ninety percent files as the input file of known $PM_{2.5}$ values.

A final set of codes was created to calculate the average error statistics between the calculated $PM_{2.5}$ values of the ten percent files before and after it was used as the validation set.

3 Results

To understand the IDW extension-based spatiotemporal interpolation formulas, hand calculations were used first. These calculations used ten made up $PM_{2.5}$ values with

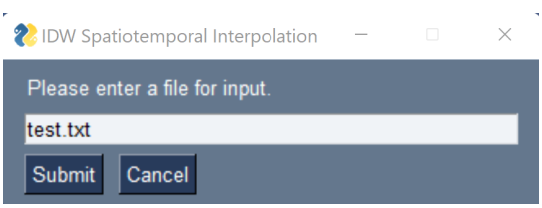
different identification numbers, x coordinates, y coordinates, and days of the year. The ten made up points are listed below.

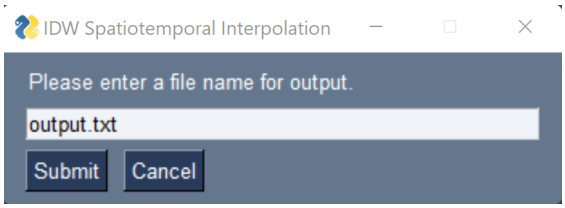
Random $PM_{2.5}$ Values						
ID	Year	Month	Day	X	Y	$PM_{2.5}$
1	2009	2	12	2	3	5
2	2009	5	20	1	1	3
3	2009	8	3	2	4	2
4	2009	6	17	3	5	1
5	2009	3	7	4	4	4
6	2009	11	5	1	1	3
7	2009	3	28	3	2	2
8	2009	4	9	5	5	4
9	2009	9	13	2	1	5
10	2009	5	20	1	2	3

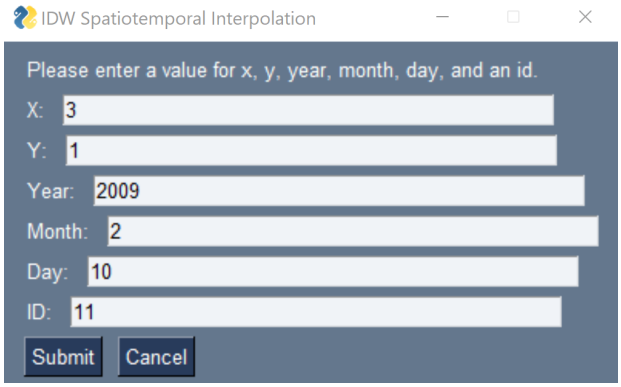
A random unknown point was picked with an identification number of 11, an x coordinate of 3, and a y coordinate of 1 on February 10, 2009. This means that the random point would be at the time dimension of 41 as February 10 is the 41st day of the year 2009. From the Euclidean distance calculations, the identifications numbers 1, 5, and 7 were the closest to the unknown point. A power value of 3 and a nearest neighbors value of 2 were selected as random numbers for the next calculations. The weights of each of the values of the closest identifications numbers were calculated. To obtain the $PM_{2.5}$ value at the unknown point, the weights were then multiplied by their known $PM_{2.5}$ values. The $PM_{2.5}$ value at the unknown location was found to be 4.973.

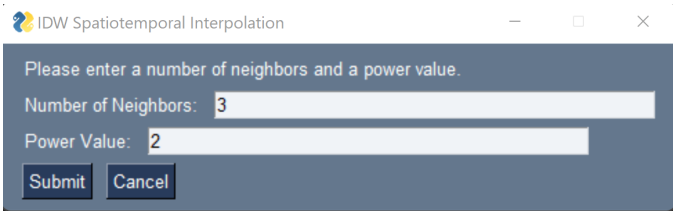
Code one was then created The first part of this code asks the user for an input file. A test.txt file was created with the ten random points listed above. This file was used as the input file. After pressing the submit button, the code then asked for an output file

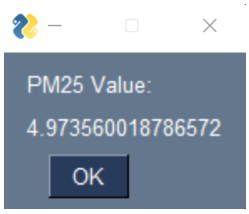
name. If the user enters a filename that does not already exist, the code creates the file and puts the output in the entered output file name. The user is then prompted to enter the unknown $PM_{2.5}$ values. The same random unknown point that was used in the hand calculation was used for testing. The user is asked for the nearest neighbors and power value. The values of 3 and 2 were used again. After all the information is submitted, the code displays a screen with the calculated $PM_{2.5}$ value. The same $PM_{2.5}$ value of 4.973 is displayed, thus meaning the math inside the code was done correctly. Finally, the user is asked if they would like to continue and interpolate another point or not. All screenshots of the prompts and interfaces are displayed below.

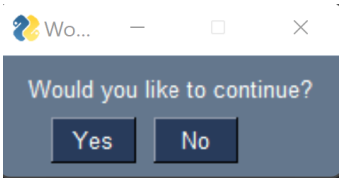




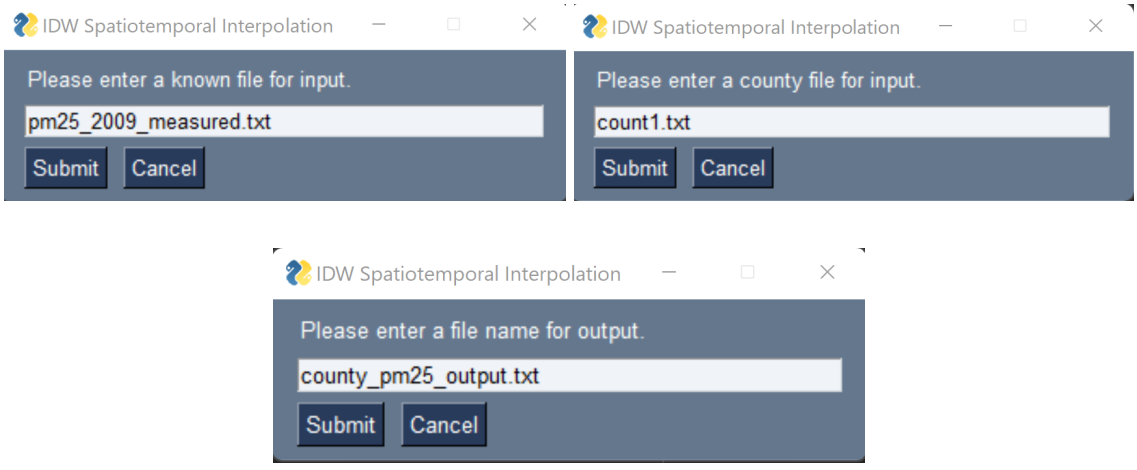








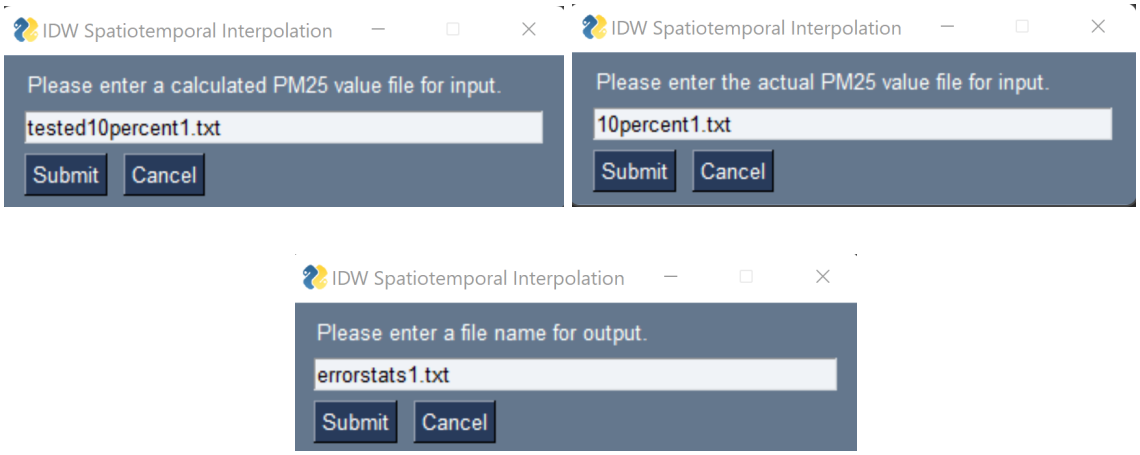
After the completion of the first code, a secondary code could be created to take another user input of an unknown list of centroid points. The user was prompted the same way as the first code for a known file input. This time the known $PM_{2.5}$ values for measured centroids throughout the United States in 2009 were used. The measured centroid file contained 146,126 known $PM_{2.5}$ values. The unknown centroid file contained 3,110 unknown locations. For each of these unknown locations, the $PM_{2.5}$ value for each day of the year in 2009 needed to be calculated, which means 1,135,150 points needed to be interpolated. The unknown centroid file had to be split up into about fifty different text files to ensure each point was interpolated as the code took about twenty minutes to compile the $PM_{2.5}$ values for the 365 days of five unknown points and slowed down with each calculation. A power value of 1 and a nearest neighbors value of 7 were used. Each calculated value for the unknown locations was added to the same output file.



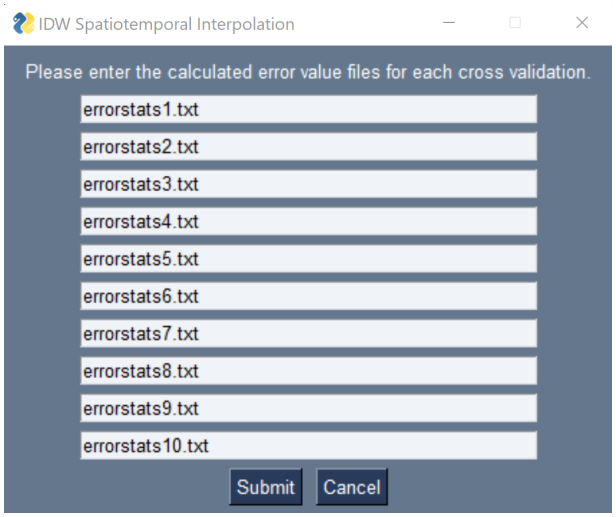
After all the unknown centroid location $PM_{2.5}$ values were placed in an output file, another set of codes could be created to split the output file for 10-fold cross validation.

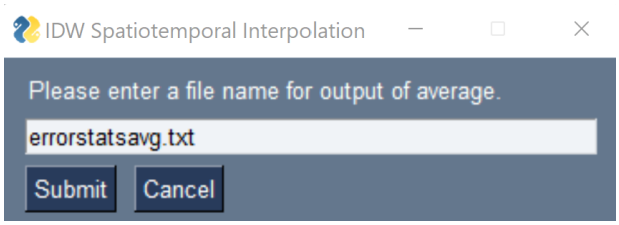
These codes split the measured centroid locations into ten files that held ten percent and ten files that held ninety percent. The ten percent files were used as the validation set, while the ninety percent files were used as the training set. The already calculated $PM_{2.5}$ value for the ten percent values were placed into a separate file to allow for the ten percent to be recalculated. Code two is then used again with the ninety percent file as the known file and the ten percent file as the county file for input. Any output file name could be chosen.

Once the $PM_{2.5}$ values are calculated again for the ninety percent, error statistics can be obtained. A final set of codes were written to convert the formulas for MAE, MSE, MARE, and RMSE into the Python programming language and take their average. The first code in this set found the error statistics for one of the folds by asking the user to enter the calculated and actual $PM_{2.5}$ value files for input. Any output file name could be entered that would contain the error statistics.



After the error statistics were gathered for each of the ten percent files, the overall average error statistics could be calculated. The second code in the set asked the user for all the ten error statistic files from the first code in this set. The user was then asked for a file output so the code could place the error statistics into a file neatly.





From the completion of all the codes, the $PM_{2.5}$ values were calculated, ten fold cross validation was completed, and error statistics were gathered. The average error statistics were completed for the time scale of c=1 with the nearest neighbors value at 7 and the power value at 1.

Error Statistics for Cross Validation $PM_{2.5}$ Calculations c = 1 (N= 7, p = 1.0)			
Mean Absolute Error (MAE)	Mean Squared Error (MSE)	Mean Absolute Relative Error (MARE)	Root Mean Squared Error (RMSE)
3.1689	61.3893	0.2630	7.6070

When comparing the error statistics above to prior research, the error statistics above are lower than most time scale options with the same nearest neighbors and power values. These error statistics are even better than time scale options with different nearest neighbors and power values [1]. The mean absolute error suggests that each $PM_{2.5}$ value

is about 3.1689 away from the true value. The mean squared error expresses that the data set is 61.3893 away from the expected line of regression, or the average of all the observations. The mean squared error highlights the outliers. The mean absolute relative error explains that the mean absolute error is only 0.2630 away from the measured $PM_{2.5}$ value. The closer this value is to zero, the more accurate the code is. The root mean squared error suggests that the code is 7.6070 off from being able to predict the correct $PM_{2.5}$ value.

4 Conclusion

The EPA measures $PM_{2.5}$ values at monitoring sites throughout the United States daily. While these values are important, some areas are not covered. This study determined the $PM_{2.5}$ values for several unknown centroid locations throughout the United States in 2009 using IDW extension based spatiotemporal interpolation. This study expanded upon the IDW extension based interpolation accuracy using a select time dimension, power value, and nearest neighbors value. The completed research also expanded upon machine learning and k-fold cross validations concepts.

This study can be helpful in determining which unmeasured centroid locations may be reaching levels that are too high. The EPA has an annual average of 12.0 $\mu g/m^3$ and a twenty four hour average of 35 $\mu g/m^3$. This means that $PM_{2.5}$ values below 12.0 $\mu g/m^3$ are considered to be healthy, while those above 35 $\mu g/m^3$ are considered unsafe. The calculated county file could be sorted to find which county $PM_{2.5}$ values are above or below these levels. From this information, the importance of measuring all counties could be stressed to the EPA and to those living in counties that may be reaching unhealthy

levels. The calculated centroids could be plotted in a map of the United States with what levels were healthy, concerning, and dangerous [13].

This study could be expanded by comparing different time scales, different power values, or different nearest neighbor values to each other. Only one time scale, power value, and nearest neighbor value was used and checked for accuracy as the Python program had limitations. The code took a longer time than expected to interpolate the centroids and, with each interpolation, the code slowed down. The code could be edited with different algorithms to prevent the limitations and speed up the interpolation process. Furthermore, the study could be expanded by using the code to interpolate unmeasured location $PM_{2.5}$ values throughout different days, months, and years.

References

[1] Lixin Li, Xiaolu Zhou, Reinhard Piltner, and Marc Kalo. 2016. Spatiotemporal Interpolation Methods for the Application of Estimating Population Exposure to Fine Particulate Matter in the Contiguous U.S. and a Real-Time Web Application. (July 2016). Retrieved October 1, 2021 from https://www.mdpi.com/1660-4601/13/8/749

[2] United States Environmental Protection Agency. 2023. Particulate Matter (PM) Basics. (February 2023). Retrieved January 20, 2023 from https://www.epa.gov/pm-pollution/particulate-matter-pm-basics#PM

[3] United States Environmental Protection Agency. 2023. Particulate Matter (PM) Basics. (February 2023). Retrieved January 20, 2023 from https://www3.epa.gov/region1/airquality/pm-human-health.html

[4] Lixin Li and Peter Revesz. 2003. Interpolation methods for spatio-temporal geographic data. (April 2003). Retrieved October 1, 2021 from https://www.sciencedirect.com/science/article/pii/S0198971503000188

[5] Lixin Li, Travis Losser, Charles Yorke, and Reinhard Piltner. 2014. Fast Inverse Distance Weighting-Based Spatiotemporal Interpolation: A Web-Based Application of Interpolating Daily Fine Particulate Matter $PM_{2.5}$ in the Contiguous U.S. Using Parallel Programming and k-d Tree. (September 2014). Retrieved October 1, 2021 from https://pubmed.ncbi.nlm.nih.gov/25192146/

[6] Lixin Li, Jie Tran, Xingyou Zhang, James B. Holt, and Reinhard Piltner. 2012. Estimating Population Exposure to Fine Particulate Matter in the Conterminous U.S.

using Shape Function-based Spatiotemporal Interpolation Method: A County Level Analysis. (January 2012). Retrieved October 1, 2021 from https://pubmed.ncbi.nlm.nih.gov/26413256/

[7] Payam Refaeilzadeh, Lei Tang, and Huan Liu. 2009. Cross-Validation. (January 2009). Retrieved October 1, 2021 from https://www.researchgate.net/publication/284400420_Cross-Validation

[8] Lixin Li, Xingyou Zhang, and Reinhard Piltner. 2006. A spatiotemporal database for ozone in the conterminous U.S. (June 2006). Retrieved October 1, 2021 from https://ieeexplore.ieee.org/document/1635995/

[9] Anne Helmenstine. 2021. Absolute and Relative Error and How to Calculate Them. (February 2021). Retrieved March 10, 2023 from https://sciencenotes.org/absolute-and-relative-error-and-how-to-calculate-them/

[10] Towards A.I. Team. 2022. 5 Regression Metrics Explained in Just 5mins. (July 2022). Retrieved March 10, 2023 from https://towardsai.net/p/l/5-regression-metrics-explained-in-just-5mins

[11] PySimpleGUI. Python GUIs for Humans. Retrieved October 10, 2023 from https://www.pysimplegui.org/en/latest/#layouts

[12] EPA. 2023. National Ambient Air Quality Standards (NAAQS) for PM. (January 2023). Retrieved March 15, 2023 from https://www.epa.gov/pm-pollution/national-ambient-air-quality-standards-naaqs-pm#:~:text=Currently%2C%20EPA%20has%20primary%20and,150%20%C2%B5g%2Fm3

[13] Lixin Li, Xingyou Zhang, and Reinhard Piltner. 2006. A spatiotemporal database for ozone in the conterminous U.S. (June 2006). Retrieved October 1, 2021 from https://ieeexplore.ieee.org/document/1635995/