Applications of Python to Evaluate Environmental Data Science Problems

Akhil Kadiyala and Ashok Kumar
Department of Civil and Environmental Engineering, The University of Toledo, Toledo, OH 43606; akumar@utnet.utoledo.edu (for correspondence)

Published online 31 October 2017 in Wiley Online Library (wileyonlinelibrary.com). DOI 10.1002/ep.12786

There is a significant convergence of interests in the research community efforts to advance the development and application of software resources (capable of handling the relevant mathematical algorithms to provide scalable information) for solving data science problems. Anaconda is one of the many open source platforms that facilitate the use of open source programming languages (R, Python) for large-scale data processing, predictive analytics, and scientific computing. The environmental research community may choose to adapt the use of either of the R or the Python programming languages for analyzing the data science problems on the Anaconda platform. This study demonstrated the applications of using Scikit-learn (a Python machine learning library package) on Anaconda platform for analyzing the in-bus carbon dioxide concentrations by (i) importing the data into Spyder (Python 3.6) in Anaconda, (ii) performing an exploratory data analysis, (iii) performing dimensionality reduction through RandomForestRegressor feature selection, (iv) developing statistical regression models, and (v) generating regression decision tree models with DecisionTreeRegressor feature. The readers may adopt the methods (inclusive of the Python coding) discussed in this article to successfully address their own data science problems. © 2017 American Institute of Chemical Engineers Environ Prog, 36: 1580–1586, 2017

Keywords: anaconda, python, pandas, scikit-learn, spyder, data science, indoor air quality, in-bus air contaminants, biodiesel, public transportation buses

INTRODUCTION

Data science may be referred to as an interdisciplinary field that helps in the extraction of significant relationships or pattern insights from available wide-ranging multi-dimensional information databases (structured, semi-structured, unstructured) through the application of relevant mathematical, computational, and scientific methods [1]. During the past decade, there has been an exponential increase in the volume of data (e.g., machine generated data based on sensors) and the type of data (e.g., people generated data based on social network comments) to be analyzed for insight generation for business as well as research purposes. Under such circumstances, it is essential that one adopts the use of the advanced analytical platforms and programming languages that can handle the large database structures of different types. With significant advancements being accomplished in the field of data science, environmental professionals are now better equipped with a set of open source software platforms that can be operated through user-friendly programming languages. This study limits itself to demonstrating the use of the Python programing language with Scikit-learn (machine learning library package) on the Anaconda platform to analyze environmental data science problems.

The Python language was created by Guido van Rossum in December 1989 at Centrum Wiskunde & Informatica in the Netherlands to serve as a successor to the ABC programming language, which handled and interfaced the Amoeba operating system. The programming language name was derived from van Rossum's favorite BBC show "Monty Python's Flying Circus." New features have been added consistently through the 90s that led to the release of Python 2.0 on October 16, 2000 and Python 3.0 on December 3, 2008. Python is a multi-paradigm programming language: object-oriented programming and structured programming are fully supported, and there are a number of language features which support functional programming and aspect-oriented programming [2]. The majority of the Python releases are general public license compatible, thereby, allowing for the flexibility of distributing a modified version without making the changes open source [3]. Anaconda is one of the world's most popular data science platform that comes with over 150 pre-installed packages, in addition to the 250 open source packages that can be installed from the Anaconda repository [4]. One may build their own custom packages and share them using the Anaconda cloud, PyPi, and other repositories. There are thousands of packages available for use from Anaconda cloud that have been developed by programmers across the world.

This study adopted the use of a comprehensive structured indoor air quality (IAQ) database obtained from performing an experimental real-world case study using the biodiesel (BD20) operated Toledo Area Regional Transit Authority (TARTA) public transportation bus (ID: 506). Details on the complete design and implementation of the field program are discussed elsewhere [5,6]. The database used in this study is the same as the database (May 8, 2007 to June 29, 2007) outlined by Kadiyala and Kumar [1,7–15]. Table 1 presents a list of all the independent variables considered to have an impact on in-bus carbon dioxide (CO_2) concentrations. The database had a total of 1,271 hourly data points with no missing values.

This study demonstrates the application of the Python machine learning library package: Scikit-learn on Anaconda platform utilizing the TARTA IAQ case study to implement a series of fundamental steps that form an integral part of any data science analytical framework. The fundamental steps include: (i) importing data, (ii) performing exploratory data analysis, (iii) performing dimensionality reduction (feature selection), (iv) developing statistical (regression) models, and (v) generating machine learning (decision trees) models. This study will prove to be valuable for managers and researches, who are inhibited by the lack of complete access to commercial software in analyzing their respective data.

INSTALLING ANACONDA (PYTHON)

Download and install the appropriate latest version of Anaconda platform (version 4.4.0) based on the operating system of the user's computer and the latest version of Python (version 3.6) from the Anaconda website [16]. Use the default options when installing Anaconda version 4.4.0 on the computer. To check for the proper installation of Anaconda, access and open the Anaconda Navigator from the Windows Start menu. One may infer that the Anaconda installation was successful when the Anaconda Navigator opens, as can be seen from Figure 1. The Anaconda Navigator is a graphical user interface that facilitates the launch of applications, manage packages, and environments without using the command-line commands. By default, the Anaconda Navigator comes with six applications: Jupyter Notebook, QTConsole, Spyder, Glueviz, Orange App, and RStudio (refer to Figure 1). Spyder is the application within Anaconda that provides a scientific Python development environment that facilitates advanced editing, interactive testing, debugging, and introspection features. Alternatively, one may use the Anaconda Prompt (accessed through the Windows Start menu) to manage applications and packages through the use of command-lines.

Table 1. Independent variable considered to influence in-bus CO_2.

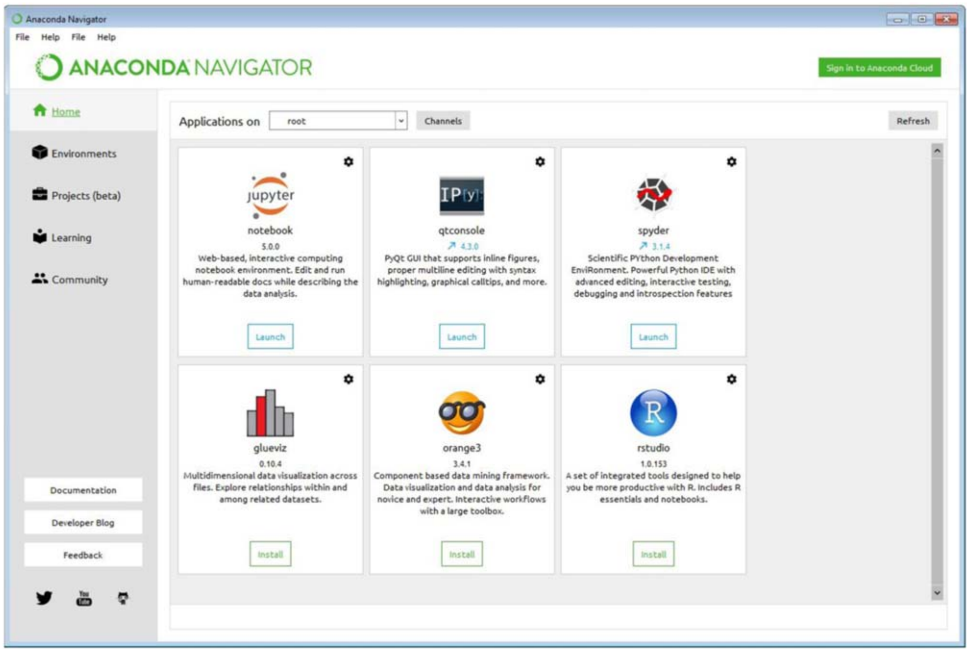
Independent Variables	
• Indoor temp.	• Indoor RH
• Ambient temp.	• Ambient RH
• Wind speed	• Precipitation
• Passenger density	• Light vehicles (Cars/SUVs)
• Bus condition/ventilation settings (run/close, idle/open, idle/close)	• Heavy vehicles (buses/trucks)

Upon successful installation of the Anaconda platform, open the Anaconda Prompt window that can be accessed through the Windows Start menu. Type the commands in the Anaconda Prompt window. To verify the successful installation of the Anaconda platform and its version, type in command 1. Use command 2 to update the Anaconda package to the latest version, if required. Command 3 provides a list of all the packages installed in the current version. One may note that all the relevant packages (Python, Pandas, Scikit-learn, Numpy, Scipy, Matplotlib) adopted for use in this study have been listed within the output from using command 3. Use command 4 to update any of the installed packages (from those listed as output from using command 3), if required within the existing working environment. Command 5 helps in determining the working environment. By default, the "Anaconda3" folder created during the installation is set to be the active root working environment. Type in command 6 to obtain the history of changes made to the working environment. Command 7 may be used to install any new packages in a new working environment. One may refer to the Conda command reference website [17] for additional spectrum of commands and their purposes to be typed into the Anaconda Prompt window.

```
> conda info ...(1)
> conda update conda ...(2)
> conda list ...(3)
> conda update PACKAGENAME ...(4)
> conda env list ...(5)
> conda list—revisions ...(6)
>conda install—name NEW-ENVIRONMENT-NAME NEW-
 PACKAGENAME ...(7)
```

IMPORTING DATA

One needs to code the Python programming in the Spyder integrated development environment (IDE) of



notebook
5.0.0
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

qtconsole
4.3.0
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

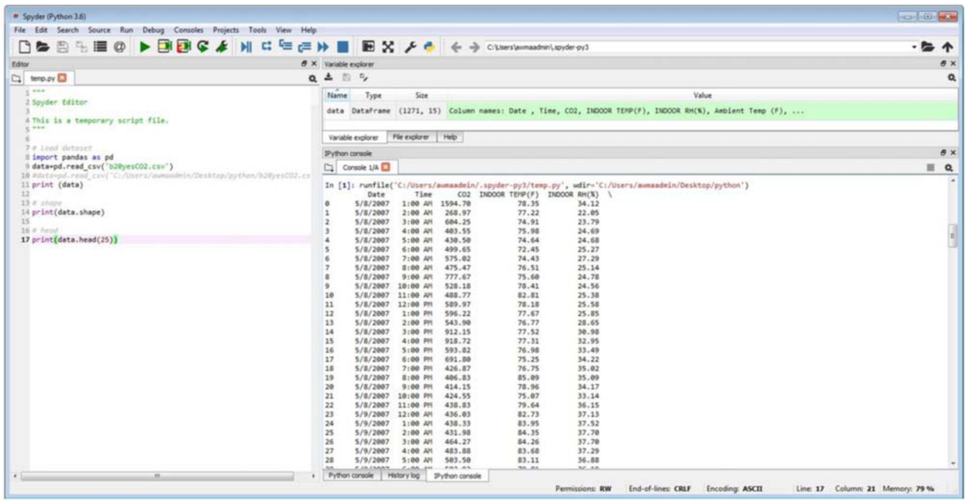
spyder
3.1.4
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

glueviz
0.10.4
Multidimensional data visualization across files. Explore relationships within and among related datasets.

orange3
3.4.1
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

rstudio
1.0.153
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

Figure 1. Screenshot of Anaconda Navigator showing the six in-built applications. [Color figure can be viewed at **wileyonlinelibrary.com**]



```
1 """
2 Spyder Editor
3
4 This is a temporary script file.
5 """
6
7 # load dataset
8 import pandas as pd
9 data=pd.read_csv('b20yesCO2.csv')
10 #data=pd.read_csv('C:/Users/awmaadmin/Desktop/python/b20yesCO2.cs
11 print (data)
12
13 # shape
14 print(data.shape)
15
16 # head
17 print(data.head(25))
```

Name	Type	Size	Value
data	DataFrame	(1271, 15)	Column names: Date , Time, CO2, INDOOR TEMP(F), INDOOR RH(%), Ambient Temp (F), ...

```
In [1]: runfile('C:/Users/awmaadmin/.spyder-py3/temp.py', wdir='C:/Users/awmaadmin/Desktop/python')
          Date      Time      CO2  INDOOR TEMP(F)  INDOOR RH(%)  \
0     5/8/2007   1:00 AM  1594.70           78.35         34.12
1     5/8/2007   2:00 AM   268.97           77.22         22.05
2     5/8/2007   3:00 AM   604.25           74.91         23.79
3     5/8/2007   4:00 AM   403.55           75.98         24.69
4     5/8/2007   5:00 AM   430.50           74.64         24.68
5     5/8/2007   6:00 AM   499.65           72.45         25.27
6     5/8/2007   7:00 AM   575.02           74.43         27.29
7     5/8/2007   8:00 AM   475.47           76.51         25.14
8     5/8/2007   9:00 AM   777.67           75.60         24.78
9     5/8/2007  10:00 AM   528.18           78.41         24.56
10    5/8/2007  11:00 AM   488.77           82.81         25.38
11    5/8/2007  12:00 PM   589.97           78.18         25.58
12    5/8/2007   1:00 PM   596.22           77.67         25.85
13    5/8/2007   2:00 PM   543.90           76.77         28.65
14    5/8/2007   3:00 PM   912.15           77.52         30.98
15    5/8/2007   4:00 PM   918.72           77.31         32.95
16    5/8/2007   5:00 PM   593.82           76.98         33.49
17    5/8/2007   6:00 PM   691.80           75.25         34.22
18    5/8/2007   7:00 PM   426.87           76.75         35.02
19    5/8/2007   8:00 PM   406.83           85.09         35.09
20    5/8/2007   9:00 PM   414.15           78.96         34.17
21    5/8/2007  10:00 PM   424.55           75.07         33.14
22    5/8/2007  11:00 PM   438.83           79.64         36.15
23    5/9/2007  12:00 AM   436.03           82.73         37.13
24    5/9/2007   1:00 AM   438.33           83.95         37.52
25    5/9/2007   2:00 AM   431.98           84.35         37.70
26    5/9/2007   3:00 AM   464.27           84.26         37.70
27    5/9/2007   4:00 AM   483.88           83.68         37.29
28    5/9/2007   5:00 AM   503.50           83.11         36.88
```

Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 17 Column: 21 Memory: 79 %

Figure 2. Screenshot of Spyder window with Editor pane (command line codes), Variable Explorer pane (imported data), and IPython console pane (showing output). [Color figure can be viewed at **wileyonlinelibrary.com**]

Anaconda by launching the application in Anaconda Navigator. Alternatively, type "spyder" in the Anaconda Prompt Window to launch the Spyder application. By default, the Spyder IDE opens with a root directory associated with the Anaconda Spyder installation folders, where the Python coded programming files can be saved. The authors have chosen to use a working directory by creating a new folder named "python" on the Desktop. The path to the working directory may be set up by providing the path information in a dialog box that opens up when one accesses the options of "Run → Configure → Working Directory" located within the Spyder application Menu Toolbar. There are multiple methods of importing different types of data into the Spyder IDE for use with Scikit-learn. More information on the loading of different types of data for use with Scikit-learn is provided elsewhere [18]. In this study, the authors already had a preformatted BD20 master sheet (in .csv format). Accordingly, the authors placed the preformatted BD20 master sheet (input data: b20yesCO2.csv) within the working directory named "python" on the Desktop. The authors have chosen to load the b20yesCO2.csv input file as a "Pandas" DataFrame. Pandas handles heterogeneous data smoothly and provides a variety of tools for manipulation and conversion into a numeric array suitable for Scikit-learn [18].

Type in the following commands in the Spyder Editor file to import the data from b20yesCO2.csv file. Type in command 8 to initiate the pandas as pd. All the subsequent references to "pd" in the Python code will call on the pandas package functions. The input data from b20yesCO2.csv data is then imported as a DataFrame and assigned to the variable named "data" by using command 9. In the absence of the input .csv file not being located within the working directory, provide the location path to the input file in the place of the name of the input .csv file. One may look at the imported data by using command 10. Type in commands 11 and 12 to obtain the shape and the headers with the top 25 rows of input data, respectively. The debug and run option for Python programming code in Spyder can be accessed through the Run Menu or by simply pressing F5 on the keyboard. The output from running the commands may be seen in the IPython console window of Spyder application. Figure 2 provides a screenshot of the Spyder (Python 3.6) window launched in Anaconda with the three integrated panes: Editor pane for coding, Variable Explorer pane for referring to the data formats, and the IPython console pane that provides the programming output.

```
import pandas as pd ...(8)
data = pd.read_csv("b20yesCO2.csv") ...(9)
print (data) ...(10)
print (data.shape) ...(11)
print (data.head(25)) ...(12)
```

EXPLORATORY DATA ANALYSIS

Generating a summary of the statistics in combination with the development of graphical visualizations for all the variables considered in any study is essential when performing exploratory data analysis. Type in command 13 to obtain a summary of statistics for all the input variables imported into Spyder. Spyder generates the statistical parameters of count, mean, standard deviation, minimum, quartile 1, median (quartile 2), quartile 3, and maximum for the imported data on using command 13. Due to space constraints, these statics have not been included in this article. To create visualization plots (e.g., box plots, histograms, etc.), one may use the "matplotlib" Python plotting library package. Type in commands 14 and 15 to ensure that Spyder calls on the "matplotlib" Python package. Commands 16 and 17 may be used for obtaining the box plots and the histograms for all the variables considered in this study, respectively. To generate box plots of one variable against another variable with labels and titles, use commands 18 through 22. Command 18 assigns the name "bp" to the generated box plot of CO_2 with passengers. Commands 19, 20, and 21 are used to assign the names for the *x*-axis, the *y*-axis, and the title, respectively. Command 22 ensures that the default tile name created when generating the box plot using command 18 is removed. Use command 23 to generate a histogram plot of CO_2 with the required number of bins in the histogram. Type in commands 24, 25, and 26 to assign the labels for the *x*-axis, the *y*-axis, and the title, respectively. Use command 27 to generate the plot.

Figure 3a illustrates the box plots of CO_2 against passengers and (Figure 3b) histogram plot of CO_2 obtained from using commands 18 through 26 in the IPython console of Spyder. From Figure 3a, one may note that the in-bus CO_2 concentrations increased generally with an increase in the passenger density. From Figure 3b, one may note that the histogram plot tended to represent a normal distribution. One may extensively develop the summary statistics and visualization plots based on the commands 13 through 27, as necessary. Due to space constraints, other plots have not been presented in this article.

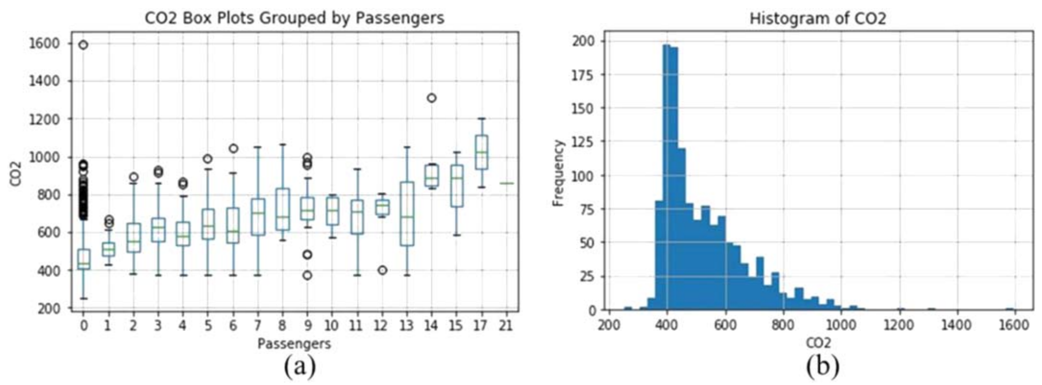


Figure 3. Graphical representation of (a) box plot for CO_2 against passenger density and (b) histogram plot of CO_2. [Color figure can be viewed at **wileyonlinelibrary.com**]

```
print(data.describe()) ...(13)
import matplotlib ...(14)
import matplotlib.pyplot as plt ...(15)
data.plot.box() ...(16)
data.hist() ...(17)
bp = data.boxplot(column=“CO2”,by=“Passengers”)
...(18)
bp.set_xlabel(“Passengers”) ...(19)
bp.set_ylabel(“CO2”) ...(20)
bp.set_title(“CO2 Box Plots Grouped by Passengers”)
...(21)
plt.suptitle(“”) ...(22)
data.hist(column = “CO2”, bins = 50) ...(23)
plt.xlabel(“CO2”) ...(24)
plt.ylabel(“Frequency”) ...(25)
plt.title(“Histogram of CO2”) ...(26)
plt.show() ...(27)
```

DIMENSIONALITY REDUCTION (FEATURE SELECTION)

Dimensionality reduction through feature selection is a process by which one may obtain a smaller subset of independent variables from a larger set of the random independent variables that capture the maximum variance in the predictor variables. In this study, there are 12 independent variables (refer to Table 1) and one target predictor variable (CO_2). There are multiple Scikit-learn processes for performing dimensionality reduction through feature selection: removing features with low variances, univariate feature selection (k-best, percentile, generic univariate), recursive feature elimination with cross-validation, feature selection using SelectFromModel (L1-based, tree-based), and feature selection as part of a pipeline [19]. The use of regression tree methods to obtain a valid subset of optimal variables for developing predictive models has been well established from prior studies by Kadiyala and Kumar [20,21]. Accordingly, this study limits itself to demonstrating the process of dimensionality reduction through the use of feature selection with tree-based SelectFromModel option.

Type in commands 28 and 29 to ensure that Spyder calls on the “RandomForestRegressor” and “numpy” Python packages. Create two DataFrames from the earlier created DataFrame (“data”): “df1” for the independent variables and “df2” for the predictor variable using commands 30 and 31, respectively. Use command 32 to create a “list” with the names of all the independent variables. Subsequently, convert the “list” into a numpy “array” using command 33. The indices for the independent variables in the numpy array as observed from the output of command 33 are as follows: feature 0—“INDOOR TEMP(F),” feature 1—“INDOOR RH(%),” feature 2—“Ambient Temp (F),” feature 3—“Ambient RH,” feature 4—“Wind Speed,” feature 5—“HourlyPrecip,” feature 6—“Passengers,” feature 7—“Cars,” feature 8—“Trucks,” feature 9—“Run/Close,” feature 10—“Idle/Open,” and feature 11—“Idle/Close,” Commands 34 and 35 may be used for calling the “RandomForestRegressor” Python package and fitting the values, respectively. Command 36 assigns the feature importance’s to the variable named “importances.” Commands 37 and 38 may be used to compute the standard deviation along the specified axis and to obtain the indices that would sort an array, respectively. Use commands 39, 40, and 41 to print the feature importances obtained from the “RandomForestRegressor” feature selection process. Type in the commands 42, 43, 44, 45, 46, and 47 sequentially to generate the plot representing feature importance’s of the forests.

```
from sklearn.ensemble import RandomForestRegressor
...(28)
import numpy as np ...(29)
df1 = data[[“INDOOR TEMP(F),” “INDOOR RH(%)”;
“mbient Temp (F),” “Ambient RH,” “Wind Speed,”
“HourlyPrecip,” “Passengers,” “Cars,” “Trucks,” “Run/
Close,” “Idle/Open,” “Idle/Close”]] ...(30)
df2 = data[[“CO2”]] ...(31)
names = [“INDOOR TEMP(F),” “INDOOR RH(%),”
“Ambient Temp (F),” “Ambient RH,” “Wind Speed,”
“HourlyPrecip,” “Passengers,” “Cars,” “Trucks,” “Run/
Close,” “Idle/Open,” “Idle/Close”] ...(32)
myarray = np.asarray(names) ...(33)
rf = RandomForestRegressor() ...(34)
rf.fit(df1, df2.values.ravel()) ...(35)
importances = rf.feature_importances_ ...(36)
std = np.std([tree.feature_importances_ for tree in
rf.estimators_], axis = 0) ...(37)
indices = np.argsort(importances)[::–1] ...(38)
print(“Feature ranking:”) ...(39)
for f in range(df1.shape[1]): ...(40)
print(“%d. feature %d (%f)” % (f + 1, indices[f], importan-
ces[indices[f]])) ...(41)
plt.figure() ...(42)
plt.title(“Feature importances”) ...(43)
plt.bar(range(df1.shape[1]), importances[indices], color-
“r”, yerr = std[indices], align=“center”) ...(44)
plt.xticks(range(df1.shape[1]), indices) ...(45)
plt.xlim([-1, df1.shape[1]]) ...(46)
plt.show() ...(47)
```

Figure 4 illustrates the RandomForestRegressor based feature importance scores obtained from using commands 42 through 47. In Figure 4, the feature importances of the

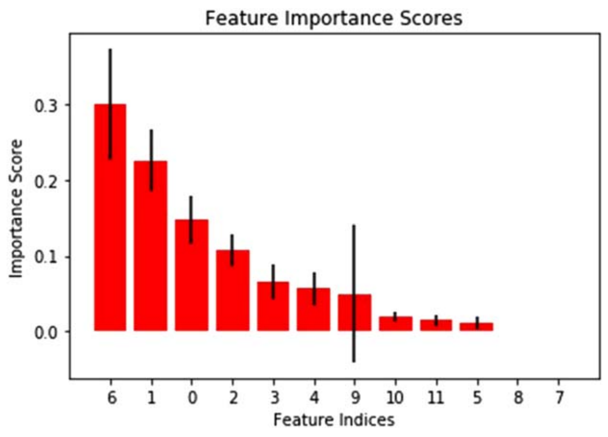


Figure 4. Graphical representation of the ranked feature selection for CO_2 obtained from using RandomForestRegressor. [Color figure can be viewed at **wileyonlinelibrary.com**]

RandomForestRegressor method are represented by the red colored bars, along with their inter-trees variability indicated by the black colored error bars. The importance scores obtained for the 12 independent variables as output for feature 0 through feature 11 are as follows: 0.148, 0.226, 0.108, 0.066, 0.057, 0.011, 0.299, 0, 0.001, 0.05, 0.019, and 0.015, respectively. Passenger density was identified to be the most important feature that influenced in-bus CO_2 concentrations (refer to Figure 4). Similar observations were made elsewhere [22,23] that identified the passenger density, indoor temperature, indoor RH, ambient temperature, and ambient RH to be the important factors influencing in-bus CO_2 concentrations when regression tree analysis was performed using the CART® software. This demonstrates that Python programming provided reliable, consistent, and comparable results in comparison with the results obtained from performing the regression tree methods using the CART® software.

STATISTICAL MODELING (REGRESSION)

Regression model development is one of the fundamental methods used widely in statistical modeling. This study demonstrates the development of a simple linear regression model for predicting in-bus CO_2 concentrations using an independent variable set selected from the application of dimensionality reduction method in the previous section of this article. The authors have chosen to develop a regression model for in-bus CO_2 concentrations on the basis of a single variable that explained the maximum variance, i.e., passenger density (refer to the results from dimensionality reduction).

Use commands 48 and 49 to ensure that Spyder calls on the relevant package functions ("linear_model," "mean_squared_error," "r2_score"). Create a new subset DataFrame of the chosen independent variables for performing the regression analysis by using command 50. The target DataFrame will remain the same as df2, considering that the predictor variable is CO_2. Split the independent variable DataFrame into training and testing sets by using commands 51 and 52, respectively. Similarly, use commands 53 and 54 to split the predictor variable DataFrame into training and testing sets, respectively. Create a linear regression model using command 55. Use commands 56 and 57 to train the linear regression model using the training sets and making the predictions using the testing sets, respectively. Type in commands 58, 59, and 60 to obtain the coefficients, the mean squared error, and the variance score. The ideal value of variance score is one. Use commands 61, 62, 63, and 64 to generate a scatter plot of the developed regression model. The coefficient, mean squared error, and variance score obtained from the development of regression model in this study are 27.68, 7544.86, and 0.16, respectively. One may add additional feature variables arbitrarily based on the dimensionality reduction method output to generate the corresponding multiple regression models. The users need to select a regression model that adequately addresses the variance in the predictor variables. Due to space constraints, further details on the additional runs were not included in this article.

```
from sklearn import linear_model . . .(48)
from sklearn.metrics import mean_squared_error, r2_score
. . .(49)
df3 = data[["Passengers"]] . . .(50)
df3_train = df3[:–20] . . .(51)
df3_test = df3[–20:] . . .(52)
df2_train = df2[:–20] . . .(53)
df2_test = df2[–20:] . . .(54)
regr = linear_model.LinearRegression() . . .(55)
regr.fit(df3_train, df2_train) . . .(56)
df2_pred = regr.predict(df3_test) . . .(57)
print('Coefficients:\n', regr.coef_) . . .(58)
print("Mean squared error: %.2f"
% mean_squared_error(df2_test, df2_pred)) . . .(59)
print('Variance score: %.2f' % r2_score(df2_test, df2_pred))
. . .(60)
plt.scatter(df3_test, df2_test, color = "black") . . .(61)
plt.plot(df3_test, df2_pred, color = "blue," linewidth = 3)
. . .(62)
plt.xticks(()) . . .(63)
plt.yticks(()) . . .(64)
```

MACHINE LEARNING (CLASSIFICATION AND REGRESSION TREES)

There have been several studies [20–28] that applied the use of machine learning methods for predicting in-bus air quality using different commercial software. This section demonstrates the application of Spyder in developing regression tree models for predicting in-bus CO_2 concentrations. The authors have chosen to develop a regression tree model for in-bus CO_2 concentrations on the basis of all the 12 independent variables considered in this study.

Use commands 65, 66, and 67 to ensure that Spyder calls on the relevant package functions ("train_test_split," "DecisionTreeRegressor," "export_graphviz"). Name the independent variable set DataFrame as "X" and the predictor variable set DataFrame as "Y" using commands 68 and 69, respectively. Split the data into training and test datasets by using command 70. In command 70, the parameter "test_size" is arbitrarily given a value of 0.3, which indicates that the test sets will be 30% of the whole dataset considered and the training sets size will be the remaining 70% of the entire dataset. The parameter "random_state" variable is a pseudo-random number generator state that needs to be used for random sampling. In order to replicate the partition results, one has to use the same value of "random_state." Create a regression tree using command 71. Kadiyala and Kumar [20,21] noted that the classification and regression trees need to have ~10% of the training data in the terminal nodes to avoid overfitting. Accordingly, the regression tree developed in this study was also developed using the stopping criterion of having a minimum of 89 ($0.1 \times 0.7 \times 1271$) samples in each of the leaf nodes. Command 72 may be used to obtain the default criteria used in the regression tree development. One may fine tune these parameters in the subsequent runs to improve the prediction accuracy or compare multiple tree model performances, if required. Use command 73 to train the regression tree model with the training sets. Obtain the accuracy on the training and test sets using commands 74 and 75, respectively. Use command 76 to make predictions using the test sets. Obtain the mean square error on the test data using command 77. Command

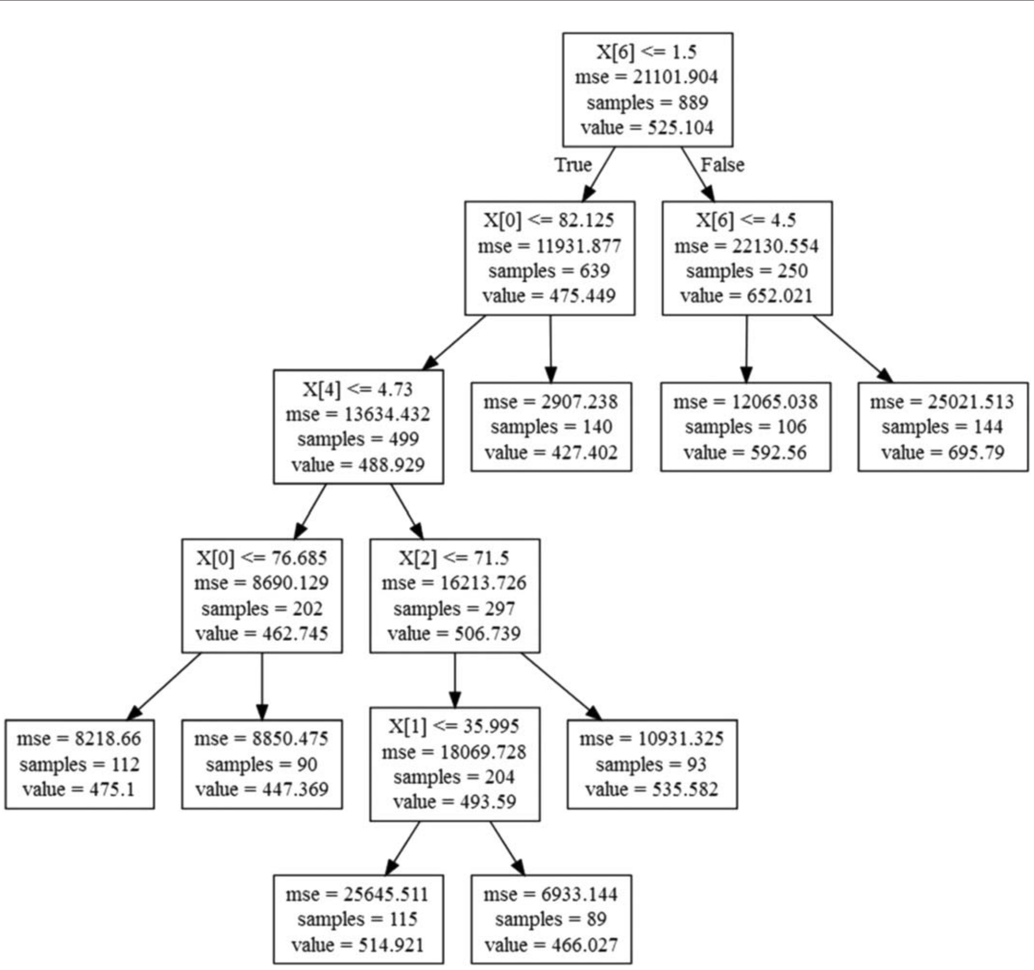


Figure 5. Regression tree developed for in-bus CO_2 using DecisionTreeRegressor.

78 may be used to save the developed regression tree output into the working directory. Use command 79 to make the predictions with test sets.

```
from sklearn.cross_validation import train_test_split ...(65)
from sklearn.tree import DecisionTreeRegressor ...(66)
from sklearn.tree import export_graphviz ...(67)
X = df1 ...(68)
Y = df2 ...(69)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.3, random_state = 100) ...(70)
regr_1 = DecisionTreeRegressor(min_samples_leaf = 89) ...(71)
print(regr_1) ...(72)
regr_1.fit(X_train, Y_train) ...(73)
print("Accuracy on training set: %f" % regr_1.score(X_train, Y_train))...(74)
print("Accuracy on test set: %f" % regr_1.score(X_test, Y_test)) ...(75)
Y_pred = regr_1.predict(X_test) ...(76)
score = mean_squared_error(Y_test, Y_pred) ...(77)
export_graphviz(regr_1, out_file="regr_1_tree.dot") ...(78)
y_1 = regr_1.predict(X_test) ...(79)
```

The developed regression tree model may be visualized by accessing the .dot extension file saved in the working directory using command 78. Open the "regr_1_tree.dot" file saved in the working directory using a text editor application. Copy the code into the Webgraphviz website [29] and click on the "Generate Graph!" button to obtain the visualization of the developed regression tree. Figure 5 illustrates the developed regression tree model for in-bus CO_2 concentrations obtained from the Webgraphviz website. The predicted CO_2 concentrations may be validated against the monitored CO_2 concentrations of the test set using operational performance measures as suggested by Kadiyala and Kumar [30,31]. Due to space constraints, the operational performance measures for the developed regression tree model have not been included in this article.

CONCLUSIONS

This study demonstrated the application of the Anaconda 4 (Spyder 3.6) software that facilitated the Python programming across the five fundamental stages of data science: (i) data importing, (ii) exploratory data analysis, (iii) dimensionality reduction through feature selection, (iv) statistical regression modeling, and (v) regression tree model development with machine learning methods utilizing the TARTA IAQ field study that monitored the in-bus CO_2 contaminants. The commands that are necessary to address the data science problems based on the Python programming language with Anaconda (Spyder) platform have been well documented in this article. It is hoped that environmental professionals will use the Python programming language methods implemented in this software review paper to gain further valuable insights from their respective databases.

ACKNOWLEDGMENTS

The authors thank the United States Department of Transportation and Toledo Area Regional Transit Authority (TARTA) for the alternate fuel grant awarded to the

Intermodal Transportation Institute of The University of Toledo. The authors express their sincere gratitude to the TARTA management and the employees for their interest and involvement in this project during data collection. The authors do not have any financial relationships with the software (Anaconda, Spyder) and the Python programming packages (Scikit-learn, Pandas, Matplotlib, Numpy, Scipy) used in this paper. The views expressed in this paper are those of the authors and do not represent the views of the funding organizations.

REFERENCES

1. Kadiyala, A., & Kumar, A. (2017). Applications of R to evaluate environmental data science problems, Environmental Progress & Sustainable Energy, 36, 1358–1364.
2. Tulchak, L.V., & Marchuk, A.O. History of python. Available at **https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/10471/461.pdf?sequence=3**.... (Accessed on September 8, 2017).
3. Python. Documentation—History and license. Available at **https://docs.python.org/2/license.html**. (Accessed on September 8, 2017).
4. Anaconda Documentation. Anaconda distribution. Available at **https://docs.anaconda.com/anaconda/**. (Accessed on September 8, 2017).
5. Kadiyala, A., Kumar, A., & Vijayan, A. (2010). Study of occupant exposure of drivers and commuters with temporal variation of in-vehicle pollutant concentrations in public transport buses operating on alternative diesel fuels, The Open Environmental Engineering Journal, 3, 55–70.
6. Kadiyala, A., & Kumar, A. (2011). Study of in-vehicle pollutant variation in public transport buses operating on alternative fuels in the City of Toledo, Ohio, The Open Environmental & Biological Monitoring Journal, 4, 1–20.
7. Kadiyala, A., & Kumar, A. (2012). Univariate time series prediction of air quality inside a public transportation bus using available software, Environmental Progress & Sustainable Energy, 31, 494–499.
8. Kadiyala, A., & Kumar, A. (2014). Multivariate time series models for prediction of air quality inside a public transportation bus using available software, Environmental Progress & Sustainable Energy, 33, 337–341.
9. Kadiyala, A., & Kumar, A. (2014). Vector time series models for prediction of air quality inside a public transportation bus using available software, Environmental Progress & Sustainable Energy, 33, 1069–1073.
10. Kadiyala, A., & Kumar, A. (2015). Univariate time series based back propagation neural network modeling of air quality inside a public transportation bus using available software, Environmental Progress & Sustainable Energy, 34, 319–323.
11. Kadiyala, A., & Kumar, A. (2015). Multivariate time series based back propagation neural network modeling of air quality inside a public transportation bus using available software, Environmental Progress & Sustainable Energy, 34, 1259–1266.
12. Kadiyala, A., & Kumar, A. (2016). Vector time series based back propagation neural network modeling of air quality inside a public transportation bus using available software, Environmental Progress & Sustainable Energy, 35, 7–13.
13. Kadiyala, A., & Kumar, A. (2016). Univariate time series based radial basis function neural network modeling of air quality inside a public transportation bus using available software, Environmental Progress & Sustainable Energy, 35, 320–324.
14. Kadiyala, A., & Kumar, A. (2016). Multivariate time series based radial basis function neural network modeling of air quality inside a public transportation bus using available software, Environmental Progress & Sustainable Energy, 35, 931–935.
15. Kadiyala, A., & Kumar, A. (2017). Vector time series-based radial basis function neural network modeling of air quality inside a public transportation bus using available software, Environmental Progress & Sustainable Energy, 36, 4–10.
16. Anaconda Documentation. Installation. Available at **https://docs.anaconda.com/anaconda/install/**. (Accessed on September 8, 2017).
17. Conda. Command reference. Available at **https://conda.io/docs/commands.html**. (Accessed on September 8, 2017).
18. Scikit-learn. Dataset loading utilities. Available at **http://scikit-learn.org/stable/datasets/index.html**. (Accessed on September 8, 2017).
19. Scikit-learn. Feature selection. Available at **http://scikit-learn.org/stable/modules/feature_selection.html#feature-selection-as-part-of-a-pipeline**. (Accessed on September 8, 2017).
20. Kadiyala, A., & Kumar, A. (2008). Application of CART and Minitab software to identify variables affecting indoor concentration levels, Environmental Progress, 27, 160–168.
21. Kadiyala, A., & Kumar, A. (2012). Examination of two different mathematical techniques for determining the important factors affecting indoor air quality. In A.R. Muñoz, I.G. Rodriguez, (Eds.), Handbook of genetic algorithms: New research (Chapter 4). New York: Nova Science Publishers, Inc.
22. Kadiyala, A., & Kumar, A. (2013). Quantification of in-vehicle gaseous contaminants of carbon dioxide and carbon monoxide under varying climatic conditions, Air Quality, Atmosphere and Health, 6, 215–224.
23. Kadiyala, A. (2012). Development and evaluation of an integrated approach to study in-bus exposure using data mining and artificial intelligence methods. Ph.D. Dissertation, The University of Toledo, Toledo, Ohio, USA.
24. Kadiyala, A., & Kumar, A. (2012). Development and application of a methodology to identify and rank the important factors affecting in-vehicle particulate matter, Journal of Hazardous Materials, 213–214, 140–146.
25. Kadiyala, A., & Kumar, A. (2012). An examination of the sensitivity of sulfur dioxide, nitric oxide, and nitrogen dioxide concentrations to the important factors affecting air quality inside a public transportation bus, Atmosphere, 3, 266–287.
26. Kadiyala, A., Kaur, D., & Kumar, A. (2010). Application of MATLAB to select an optimum performing genetic algorithm for predicting in-vehicle pollutant concentrations, Environmental Progress & Sustainable Energy, 29, 398–405.
27. Kadiyala, A., Kaur, D., & Kumar, A. (2013). Development of hybrid genetic-algorithm-based neural networks using regression trees for modeling air quality inside a public transportation bus, Journal of Air &Waste Management Association, 63, 205–218.
28. Kadiyala, A., & Kumar, A. (2013). Artificial intelligence: Emerging approaches for environmental data analysis, EM (*by A&WMA*), August Issue, 4–7.
29. Webgraphviz. WebGraphviz is graphviz in the browser. Available at **http://www.webgraphviz.com/**.(Accessed on September 8, 2017).
30. Kadiyala, A., Kumar, A. (2012). Guidelines for operational evaluation of air quality models (123 pp). Saarbrücken, Germany: LAP LAMBERT Academic Publishing GmbH & Co. KG. ISBN: 978-3-8465-3277-5,.
31. Kadiyala, A., & Kumar, A. (2012). Evaluation of indoor air quality models with the ranked statistical performance measures using available software, Environmental Progress & Sustainable Energy, 31, 170–175.