A close-up of a car

Description automatically generated

EV Charging

Fleet monitoring

Amber J Lee, Marie Claire O’Connell, Paul C Vose

| IMT 575 A Au 24: Data Science III: Scaling, Applications, And Ethics | 11/24/2024

# Table of Contents

[Executive Summary 2](#_Toc183371487)

[Introduction 2](#_Toc183371488)

[Machine Learning Models 3](#_Toc183371489)

[Deep Learning Models 5](#_Toc183371490)

[Initial Iteration of FCNN with basic hyperparameter tuning 6](#_Toc183371491)

[Second iteration of fcnn with hyperparameter tuning 9](#_Toc183371492)

[third iteration of fcnn with additional hyperparameter tuning 13](#_Toc183371493)

[Conclusion 18](#_Toc183371494)

[Bibliography 18](#_Toc183371495)

# Executive Summary

By using a publicly accessible dataset that specifies various features about Electric Vehicles that was recorded by onboard data loggers, we created various models that can be used to provide predictions to help optimize EV usage and educate users on how much electricity their EV can use. Specifically, we will use various features such as total distance driven, driving time and others to train machine learning models to predict Regenerated Energy to assist EV users on how to optimize charging for their Electric Vehicles. A Linear Regression model and a Gradient Boosting Regression model were created to predict this feature, and both models proved effective in their predictions. Additionally, we were also curious if we could use similar features to predict how much electricity an Electric Vehicle would use. We opted to train a Random Forest Regression model and a Fully Connected Neural Network (FCNN) to predict the Total Energy Consumed (kWh) by an Electric Vehicle. Our first FCNN model had very little hyperparameter tuning and served as proof of concept that an FCNN model would be effective in predicting the target variable. After this model was shown to be effective, we created two additional models with additional hyperparameter tuning to increase its capability. We were able to show that by further tuning the hyperparameters for the FCNN, we were able to literately reduce error via the mean absolute error (MAE) metric. Of the three FCCN models that were made, each model produced a lower MAE than the previous one. Lastly, the Random Forest Regression was also shown to provide useful predictions for Total Energy Consumed (kWh). The potential we envision for these models is twofold. Firstly, they can enhance Battery Management by estimating energy consumption, predicting remaining battery life and ensuring more efficient battery usage. Lastly, they can optimize Electric Vehicle use by providing insights into driving strategies that minimize energy consumption, such as recommending optimal speeds, acceleration profiles, and other driving behaviors.

# Introduction

The data used in this analysis was obtained from the **DOE EV Data Collection - Vehicle Data** via [Livewire Energy](https://livewire.energy.gov/ds/calstart/vehicle). This dataset consists of electric vehicle performance data collected directly from the vehicles during standard operations. The data was gathered using onboard data loggers, which were either installed by the initial project team or pre-installed by the original equipment manufacturer. There are 47 different features in this dataset, and they range from the numbers of trips taken by an EV during a day, to the average GPS speed that was logged by the onboard data logger. To view all the features, we invite you to click the link above to explore the dataset. The features are also clearly displayed under the **Data Exploration and Cleaning** section of our Jupyter Notebook that is attached separately.

We identified two areas of interest:

**How can we go about optimizing battery usage for EV users?**

* A **Gradient Boosted Regression** model will be designed to display how we can predict Regenerated Energy based on features in the dataset. Through testing, the most significant features will be identified.

**Can we accurately predict how much electricity will be used by EVs?**

* We will use **Random Forest** R**egression** to predict the total energy consumed by EVs based on features within the data set.
* A **Fully Connected Neural Network (FCNN)** will identify patterns between various factors and total energy consumed by EVs. This will provide a more holistic understanding of how users can optimize their EV performance.
* We will **refine** our FCNN model through multiple stages of hyperparameter tuning to create the most effective version of the model.

# Machine Learning Models

To better understand electric vehicle charging patterns, we identified two variables of interest within our data: total energy consumption and regenerated energy. These two variables were found to have a significant relationship to one another with a reported correlation of 0.82. This is further supported by the following graph which displays the results of a linear regression model of the two variables as the points are grouped near the line of best fit.

A red line with blue dots

Description automatically generated

While it is evident that total energy consumption and regenerated energy produced by electric vehicles are related, we proceeded to examine what driving patterns were most influential on these variables. Through some trial and error, idling time, driving time, state of charge (SOC) used, average speed, and total distance proved to be quite indicative of an electric vehicle’s total energy consumption. We built a random forest regression model to explore the relationship, and it was found to have a coefficient of determination of about 0.93, meaning that about 93% of the variation in total energy consumption could be explained by the five mentioned variables. The below visualization displays the true total energy consumption values compared to the predictions determined by the random forest model. The different categories have a high overlap which further cements that the model can make predictions in total energy consumption that we can be confident are near the true value.

A graph of energy consumption

Description automatically generated

These same five variables—idling time, driving time, SOC used, average speed, and total distance—were also found to be key to predicting the regenerated energy produced by an electric vehicle. A gradient boosting regression model was developed to determine the ability of these variables to find regenerated energy, and the model achieved a coefficient of determination of about 0.95. This score is very close to one, meaning that most of the variation within the regenerated energy values can be attributed to those five variables. Again, the below plot displays the predicted values against the true values of regenerated energy, and there is a very high overlap between them.

A graph of different colored bars

Description automatically generated with medium confidence

Variables that are easily tracked by electric vehicle users have proven to be especially indicative of how much energy is consumed and generated by electric vehicles. This suggests that changes to any of these factors will have a direct and measurable impact on overall energy usage and regeneration.

# Deep Learning Models

Fully Connected Neural Networks (FCNNs) offer a powerful approach for predicting energy consumption in electric vehicles (EVs) by modeling the complex relationships between various features, such as vehicle specifications, driving conditions, and environmental factors. These models can incorporate a wide range of input variables, including vehicle parameters like weight, motor efficiency, and battery size, along with driving data such as speed, distance traveled, and road type. FCNNs can also account for external factors, such as temperature and wind speed, which impact energy efficiency. By training an FCNN with data from EV usage, the model learns to predict energy consumption, which can be essential for applications such as route planning, battery management, and optimizing driving behavior. With their ability to handle large datasets and capture nonlinear relationships, FCNNs can provide accurate predictions of energy usage, aiding in more efficient energy management and longer battery life for electric vehicles.

**Purpose: To see whether we can accurately predict Total Energy Consumption by an Electric Vehicle when using the features provided in our dataset. By accurately predicting Total Energy Consumption, we can easily translate this to predicting the electricity that is used by an EV.**

## Initial Iteration of FCNN with basic hyperparameter tuning

**Model:**





* Architecture:

Above, we can see our first iteration of a Fully Connected Neural network model that is used to predict Total Energy Consumption using all of the features provided in the dataset.

The first layer of this model is the input layer which includes 256 Neurons, a Rectified Linear-Unit which allows the model to learn non-linear patterns, and the number of input features which includes all features in the data except the target feature. Lastly, we include BatchNormalization to stabilize training through normalization of the layer activations and follow with dropout which randomly regularizes 50% of the Neurons in training to prevent overfitting.

The second, third and fourth layer follow a similar construction of the input layer above. However, these three layers are known as Hidden layers which iteratively refine what has been learned from the previous layers. The reason these layers are called ‘Hidden layers’, is because their activations cannot be directly observed in either the input or output of the neural network. In the model, each hidden layer progressively decreases in the number of neurons and dropout rate. The rationale for this design decision was that this would allow us to capture broad patterns in the features in our early layers, which could then be iteratively refined in the layers that come later in the model.

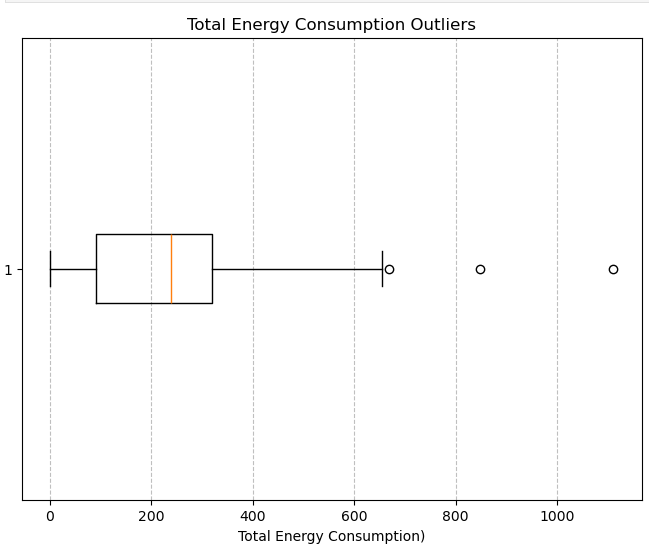
Lastly, we have our activation layer where a single neuron will output a continuous value which is the prediction. Since our prediction variable is continuous in nature, our prediction will be as well. Using a regression activation function in the final layer will allow us to accommodate this need. There are other activation functions such as Sigmoid, which lends itself useful for binary classification, or SoftMax which is ideal for multi-class activation, but neither are well suited as regression for this particular task.

The model is then compiled by specifying the optimizer, loss function and metrics. The optimizer in this case is the Adam optimizer, which is a variant of gradient descent. This means the model will iterate through different weights until eventually it hits the global minimum for loss, also known as error for our predictions, and the results will indicate the best weights to be used for the model. The loss function chosen for this model is Mean Squared Error (MSE), and it will determine what the loss is as the model progresses through iterations. Additionally, we will use Mean Absolute Error (MAE) as our metric. This will not have a direct effect on the model but will produce a value that will be easier for us to interpret when investigating how well the model is performing. Lastly, we specified the use of 45 Epochs with a batch size of 32. We can think of this as the model processing the data in set number of batches per Epoch- each epoch is essentially one iteration of the model.

* Elaboration on Hyperparameter Tuning:

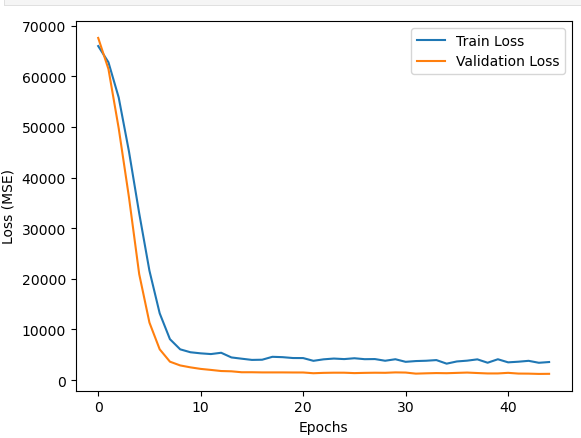
Regarding the hyperparameters and architecture used, our first iteration of our Fully Connected Neural Network is mostly based on our pre-existing thoughts and assumptions about what would be the most effective model. In this iteration, most of our testing was to find the most ideal number of hidden layers, neurons, epochs and drop-out rate percentage. We initially began with one hidden layer but noticed a significant drop in MAE when adding two more and increasing the initial neuron count, but we had an increase in MAE if we added more layers. It seemed as though having 3 hidden layers for this model was the most ideal. Similarly, we found that 45 Epochs and a Batch size of 32 was the most effective, these parameters led to the lowest loss and increasing or decreasing them led to an increase in loss. Furthermore, we opted for the Adams optimizer and the MSE loss function due to their common use and effectiveness for regression tasks. We will iterate through other parameters, including learning rate, in future models to see if we can further decrease loss.

* Removing Outliers:

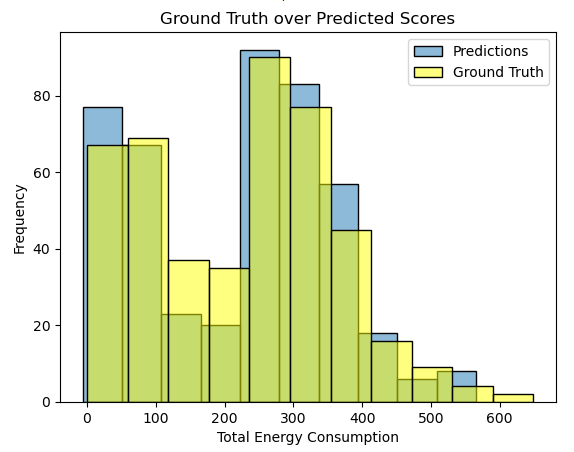


One notable improvement we made to this model was removing outliers in the target variable. As we can see above, there are significant outliers that are above 800 kWh for Total Energy Consumed. After dropping these outliers, we had a noticeable improvement in MAE as the error decreased.

**Evaluation of Model:**



In the graph above, we can see both the training loss and validation loss (MSE) across all the epochs of the model. Both plateau around 8 epochs and then remain constant for the remainder of epochs. In the real world, this would make a good argument to decrease the number of epochs in the model to preserve computational resources, but this was kept the same for later iterations of the model in order to provide better comparisons. It is also notable that the validation loss becomes constant before the training loss, suggesting that our model performs better on the testing dataset than the training dataset.



When comparing the predictions and ground truth comparisons, it becomes clear that our model performs reasonably well. Above, we can see the overlap between the ground truth values (yellow) and the predictions from our model (blue). It is evident that the distribution for each is very similar, and they both share relatively the same shape. The graph above also reveals that our model tends to slightly underestimate the ground truth values.

MAE: It terms of our chief metric; this model produced an MAE of 21.1931.

## Second iteration of fcnn with hyperparameter tuning

One of the most crucial steps in model development is hyperparameter tuning, where the objective is to find the optimal settings that allow the model to perform well on unseen data. Hyperparameter tuning plays a critical role in enhancing model accuracy, generalization, and efficiency. One of the most effective methods for hyperparameter optimization is **RandomizedSearchCV** as this technique enables the search for the best set of hyperparameters by randomly sampling combinations from a predefined grid of values, ensuring a balance between efficiency and thoroughness.

**Model:**

* Hyperparameter Tuning:

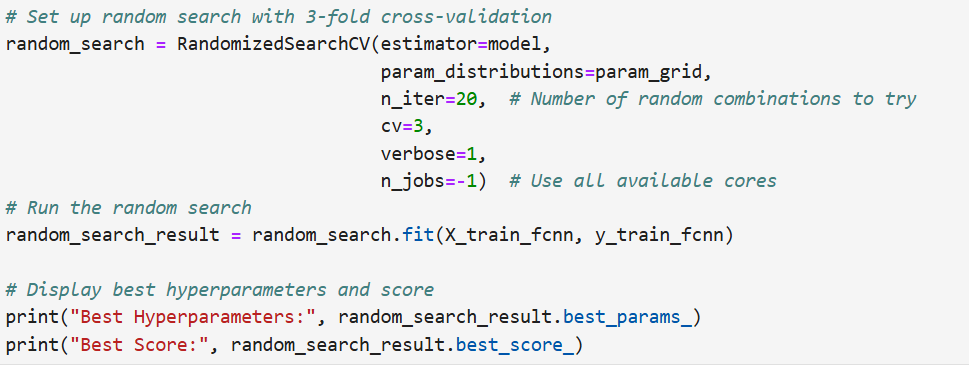
The architecture of the Fully Connected Neural Network (FCNN) in this code is designed for regression tasks, particularly predicting a continuous target variable (Total Energy Consumption). The model has an input layer followed by four hidden layers with decreasing numbers of neurons to gradually reduce dimensionality. The input layer is tailored to the number of features in the dataset. The first hidden layer has a configurable number of neurons (e.g., 64 by default), with ReLU activation to introduce non-linearity. Each hidden layer employs **Batch Normalization** to stabilize and accelerate training, and **Dropout** for regularization to mitigate overfitting. The subsequent layers progressively reduce neuron counts by half, quarter, and eighth of the initial neurons, applying a similar pattern of ReLU, Batch Normalization, and Dropout. The output layer has a single neuron with a linear activation function, appropriate for regression tasks.

Hyperparameter optimization is performed using a **Randomized Search Cross-Validation** with a KerasRegressor wrapper for compatibility with scikit-learn's grid search utilities. The key hyperparameters being optimized include the **optimizer**, with options like Adam, SGD, RMSprop, and Nadam, each tuned with different learning rates (0.001, 0.0005, 0.0001). The number of neurons in the first hidden layer varies between 32 and 512, and dropout rates range from 0.2 to 0.8. Training-specific parameters, such as batch sizes (16, 32, 64, 128) and epochs (10, 20, 50, 100, 150, 200), are also included in the search. The random search evaluates 20 combinations of these hyperparameters using 3-fold cross-validation to find the best configuration based on performance metrics like mean absolute error (mae) and mean squared error (mse).

This setup aims to balance model complexity and generalization, leveraging random search to efficiently navigate the large hyperparameter space. With techniques like Batch Normalization and Dropout, the architecture is robust to overfitting and effective in handling varying dataset scales. The selected hyperparameters optimize both the learning process and the model's capacity to capture complex relationships in the data.

Best Hyperparameters: {'model\_\_optimizer': 'sgd', 'model\_\_neurons': 128, 'model\_\_learning\_rate': 0.0001, 'model\_\_dropout\_rate': 0.3, 'fit\_\_epochs': 100, 'fit\_\_batch\_size': 64}

Best Score: 0.9902037972591803



* Architecture:

The architecture is a copy of the basic model used in the **Randomized Search.** The architecture is coupled with an automated hyperparameter optimization process that selects the best combination of learning rate, optimizer (e.g., Adam, SGD, RMSprop), dropout rates, batch size, and epochs based on cross-validation results. This dynamic approach ensures the model remains efficient, robust, and adaptable to changes in the data, striking a balance between complexity and regularization. The architecture’s modularity and automation make it highly effective for regression tasks in evolving environments.

A screenshot of a computer program

Description automatically generatedA screenshot of a computer code

Description automatically generated

**Evaluation of Model:**

A graph of a graph

Description automatically generated

The plot indicates that the model has been well-trained, with both the training and validation loss steadily decreasing and closely following each other throughout the epochs, which suggests good generalization and no overfitting. The losses stabilize around epoch 10, indicating that the model has likely converged. The final loss values (approximately 5,000 or lower) demonstrate that the model has achieved a reasonable fit to the data. Overall, the model appears to perform well, but further evaluation using additional metrics and unseen test data is recommended to confirm its real-world applicability.

. A graph of a graph of a graph

Description automatically generated with medium confidence

This histogram compares the predicted values to the ground truth for total energy consumption. The yellow bars represent the ground truth distribution, while the blue bars overlay the predictions made by the model. The alignment of the predictions with the ground truth is relatively close in many regions, except around the 200–300 range, where the model captures the peak of the distribution. However, there are discrepancies in some areas—for instance, the model underpredicts lower values (below 100) and overpredicts higher values (above 500), as indicated by the mismatched bar heights in these regions.

A graph with purple dots

Description automatically generated The correlation graph above indicates that the model tends to exhibit the greatest errors in the lower and higher ranges. This is likely attributable to the data distribution, which is skewed to the right.

MAE: This model achieved a Mean Absolute Error (MAE) of 13.6267.

## third iteration of fcnn with additional hyperparameter tuning

We realized that our initial data frame might not meet our needs due to the size of the data set after dropping all rows with one NA. We revisited the data loading process and decided to focus on using fewer features with more reliable data to ensure greater data integrity.

To handle missing values within the dataset, we employed a series of steps to ensure the integrity and completeness of the data. To recover from this reduction and obtain a more substantial dataset, we then chose to retain the missing values as NAs and focused on selectively addressing them. By utilizing logic, we create a few different code segments to address the missing data. We sorted by "Vehicle ID" and "Date" in ascending order. To account for some of the missing values in two columns, "Initial Odometer" and "Final Odometer” we were able to fill forward and backwards using the shift(1) method to ensure which column we obtained the data from. For example, in the "Initial Odometer" column, any missing values are filled by the "Final Odometer" value from the previous row within the same "Vehicle ID" group. These steps help to maintain continuity in odometer readings for each vehicle while addressing gaps in the data. We did this for all the features we could. We also dropped duplicate rows and rows with NAs.

The Hyperparameter Tuning and Architecture was set up in the same manner as the second model. The **Randomized Search** resulted in

Best Hyperparameters: {'model\_\_optimizer': 'adam', 'model\_\_neurons': 256, 'model\_\_learning\_rate': 0.0005, 'model\_\_dropout\_rate': 0.6, 'fit\_\_epochs': 50, 'fit\_\_batch\_size': 128}

Best Score: 0.9401962004645904

**Evaluation of Model:**

A graph of a line graph

Description automatically generated

The plot indicates that the model has been well-trained, with both the training and validation loss steadily decreasing and closely following each other throughout the epochs, which suggests good generalization and no overfitting. The losses stabilize around epoch 15, indicating that the model has likely converged. The final loss values (approximately 1,000 or lower) demonstrate that the model has achieved a reasonable fit to the data. Overall, the model appears to perform well, but further evaluation using additional metrics and unseen test data is recommended to confirm its real-world applicability

A graph of a graph

Description automatically generated with medium confidence

In the graph above you can see that the data is skewed but matched with the original histogram. Improvement can be made to the model by addressing the distribution of the data. However, the model seems to capture the true data.

**A graph showing a number of dots

Description automatically generated with medium confidence**

The correlation graph above aligns with the prior graph showing that there is a greater error since the model distribution is skewed to the low energy consumption levels.

**Model Improvements &** **Reevaluation:**

To deal with the distribution model we applied a log transformation to the target variable to reduce skewness y = np.log1p(y). After running the models again we obtained Best Hyperparameters: {'model\_\_optimizer': 'nadam', 'model\_\_neurons': 256, 'model\_\_learning\_rate': 0.0005, 'model\_\_dropout\_rate': 0.5, 'fit\_\_epochs': 150, 'fit\_\_batch\_size': 64} Best Score: 0.9677305710083782.

A graph with a line graph

Description automatically generated with medium confidence

Because the model uses a linear activation function in the output layer for a regression problem, it may predict negative values even when the target range is strictly non-negative. This can happen because the model isn't constrained to produce positive outputs. We addressed this issue by applying a **clip operation** to constrain predictions during evaluation. y\_pred\_FCNN\_clipped = np.clip(y\_pred\_FCNN.flatten(), 0, None)

A graph of different colored bars

Description automatically generated with medium confidence

A graph showing a line of dots

Description automatically generated with medium confidence

These changes resulted in a more accurate model with fewer prediction errors. Alternatively, replacing the final linear activation with a ReLU function can constrain the output to be non-negative, yielding similar results. However, we opted to retain the linear activation for this implementation.

A graph of a train loss

Description automatically generated

Upon reviewing the model's loss, we implemented early stopping to enhance performance and prevent overfitting. Below is the image of the code addition.

A close-up of a message

Description automatically generatedA graph of a graph

Description automatically generated

This modification helps the model stop training once it has converged, avoiding unnecessary epochs and potentially enhancing generalization performance

A graph of different colored bars

Description automatically generatedA graph showing a line of purple dots

Description automatically generated

While this modification did not significantly impact the model's error rates, it did influence the frequency of the predictions.

The final MAE: This model achieved a Mean Absolute Error (MAE) of 0.2198.

Ideally having all the feature data from the initial data set would produce a more realistic model. For example, having the regeneration energy and the average speed would make a huge difference in the last model.

# Conclusion

The random forest and gradient boosting regressors provide valuable insights into the effects of driving patterns on energy consumption and regeneration. Combined with the trained FCNN model, there is significant potential for predicting the energy consumption of electric vehicles (EVs) under various driving conditions. In this demonstration, all our ML models and each iteration of our FCNN performed quite well. This can be seen by the strong overlap between the predictions our models made, and the ground truth values from the testing data. Regarding our FCNN, our last iteration proved to have the lowest error and is therefore the most effective. For instance, our first FCNN produced an MAE of 21.19, our second an MAE of 13.62, and our last iteration had an MAE of 0.2198. The capability of our models, both ML and FCNN, can be applied to several critical tasks, including:

* **Battery Management**: By estimating energy consumption, the models can help predict the remaining battery life, ensuring more efficient battery usage.
* **Optimization**: The models can provide insights into driving strategies that minimize energy consumption, such as recommending optimal speeds, acceleration profiles, and other driving behaviors.

By training the FCNN model on comprehensive data, we can develop a robust solution that accurately forecasts energy consumption, enabling users to optimize their driving habits and better plan for charging needs.

The use of **Randomized Search Cross-Validation** for hyperparameter optimization is an essential practice, enhancing model performance while saving time and allowing for more scalable AI solutions.

To truly assess the model's effectiveness, it is crucial to deploy it in a production environment. Doing so would enable continuous performance monitoring and allow for data collection to further refine and enhance the model over time.

# Bibliography

U.S. Department of Energy. (n.d.). *Livewire data platform*. Retrieved 2024, November 24th, from <https://livewire.energy.gov/ds/calstart/vehicle>

ChatGPT. (2024, Noverber 24). Improving writing style for technical and academic explanations. https://chat.openai.com.

Chollet, F. (2021). *Deep Learning with Python, Second Edition.* Manning Publications.