THE UNIVERSITY OF CHICAGO

A Comparison of Methods to Predict Multisite Gasoline Prices

By

Haylee Ham

June 2018

A paper submitted in partial fulfillment of the requirements for

the Master of Arts degree in the Master of Arts in

Computational Social Science

Faculty Advisor: Dr. Luc Anselin

Preceptor: Ryan Hughes

Abstract:

The goal of this study is to test the accuracy of three forecasting methods on a multisite time series prediction problem. Multiple linear regression, random forest, and long short-term memory (LSTM) network models were used to predict the price of gasoline at each of 12,374 stations across Germany every day for 30 days. I found that the random forest model outperformed the multiple linear regression and LSTM network models on average across all stations and days.

1. Introduction

The ability to accurately forecast gasoline prices has been of interest as it is an important factor in determining the future demand for gasoline, automobiles and other vehicles that require gasoline (Allcott et al. 2014), the potential success of investments in energy-related innovations (Anderson et al. 2011), and the effect of gasoline taxes on carbon emissions (Alquist et al. 2011). These studies have aggregated gasoline prices to the national level rather than attempting to predict the gasoline price at each individual station. Consumers are directly affected by the pricing decisions made by local stations and their behavior about whether and how much to consumer is affected by the national trend of gasoline prices as well as retail gasoline pricing behavior of local stations (Borenstein et al. 1993).

Inferential models have been developed that attempt to explain heterogeneous gasoline prices at each station in a given dataset (Borenstein et al. 1993, Shepard 1993, Hosken et al. 2008, Kvasnička et al. 2018, Satoh et al. 2018, Haucap et al. 2018).

However, this study will create and train models that predict the varying prices at individual stations across a country. In particular, the models are created to predict daily gasoline price at each of 12,374 gasoline stations across Germany (roughly 86% of all gasoline stations in Germany) 30 days in the future.

In this study, I will compare the accuracy of a linear regression model, a random forest model, and a long short-term memory network trained to predict gasoline prices at each station.

A random forest is made up of many regression trees that are each fit to a sample of the data. When grown sufficiently deep, a tree is able to “capture complex interaction structures in the data” (Hastie et al. 2001). However, trees are generally very noisy since the predictions each tree makes is highly dependent upon the sample it is trained on. By combining these regression trees into the ensemble of a random forest, this noise can be quieted by averaging the prediction from each tree into one low-bias, low-variance prediction (Hastie et al. 2001).

A long short-term memory (LSTM) network is a type of recurrent neural network (RNN). A recurrent neural network is a network whose “underlying topology of inter-neuronal connections contains at least one cycle” (Medsker et al. 2000), meaning that the output of the network can be used as an input in the next iteration of prediction. An LSTM network solves problems commonly faced by RNNs, such as vanishing and exploding gradients and is designed to learn long-term dependencies found in time series data (Sak et al. 2014). An LSTM network contains an input layer, an output layer, and one or more hidden layers that contain the characteristics that allow it to remember time-series dependencies (Fischer et al. 2017).

Common multiple time series methods are LSTM networks (Zang 2017), random forests (Lin 2017), and vector autoregression (Pravilovic 2017). Vector autoregression is a generalization of the autoregressive model that describes time-varying processes by being linearly dependent on previous values of the dependent variable. Vector autoregression allows this to be applied to multiple time series (Pravilovic 2017). However, this dataset contains a much larger cross-sectional dimension than time-series dimension, which means that the VAR model is not an appropriate approach (Bond 2002).

To the best of my knowledge, this is the first time that prediction models have been applied to multisite gasoline price prediction.

2. Literature

In an effort to predict gasoline prices across a large geographic area, it is important to first note the forces at work in the setting of local gasoline prices. First, the spatial component of how gasoline prices are set in relation to prices at nearby stations should be considered. Various studies have addressed the question of the spatial relationship of gasoline prices. Robert Haining's 1984 paper entitled "Testing a Spatial Interacting-Markets Hypothesis" analyzed gasoline prices in Sheffield, England. He discussed the differences in price patterns in urban vs non-urban areas. He found that gasoline price in urban areas, with customers who can collect relative information on gasoline prices and travel to other retailers at almost no cost to themselves, set costs that are more correlated to neighboring stations than in non-urban areas, where markets do not overlap to such a great degree. Haining also found that during times of price cuts (a signal of competition) there is a significant amount of spatial clustering on the lowest end of gasoline prices. On the other hand, Haining found that in times of rising prices, localized competition became weaker and interaction effects were not linked at any price level. This suggests that spatial patterns are more influenced by the relative location of stations and other inter-market interactions when prices are falling (Haining 1984). It has also been confirmed that spatial patterns exist in the decision of gasoline prices and that local markets are highly influential, eschewing the theory that commuters smooth the spatial patterns by spreading their demand beyond local markets (Eckert et al. 2004). Due to this spatial relationship, it will be important to include the latitude and longitude of all stations in the models so that information about the proximity of one station to another can be used to inform predictions.

Haining also found that prices increased as a station's distance from major roadways increased (Haining 1984). For this study, a dummy variable that indicates whether a station lies directly beside the German Autobahn will be included in the models along with a variable that measures the distance between the Autobahn and the station. These variables will allow the model to capture this relationship between proximity to a major roadway and price.

Retail gasoline prices are not only dependent on station-specific variables, but are also affected by temporal circumstances. A study conducted by Andrew Eckert et al.

focused on price cycles of gasoline. They were specifically interested in attempting to identify collusion and predatory behavior in setting gasoline prices in Vancouver, Canada. Their work found that large increases in price do not happen when prices fall too close to the wholesale price, but rather on days of the week when demand is lowest. These increases begin in Vancouver and then move east. However, price decreases happen more quickly in the east, where the lowest concentration of population exists. This is consistent with previous findings of Edgeworth cycles, where price decreases begin in local areas where most competitors are clustered and then move across the entire market (Eckert et al. 2004). Variables identifying which state the station resides in will be included in the model, as well as the aforementioned latitude and longitude coordinates of each station. These variables may be helpful in discerning low population concentration in certain states.

A publication entitled, “Do Gasoline Prices Respond Asymmetrically to Crude Oil Price Changes?” found a stronger relationship between increases in crude oil and gasoline prices as compared to decreases in crude oil and gasoline prices. Thus, there is an asymmetric response between crude oil prices and prices of gasoline (Borenstein et al. 1997). In order to capture this pattern, both the wholesale price of gasoline and oil price are included in the model such that the models may judge the relationship geographic areas or stations have with these prices.

The ability to measure these observed patterns is highly dependent upon the granularity of the data available. Eckert et al. seek to identify which phenomena would be incorrectly measured or lost entirely depending on the type of data used to study these price cycles. They found that most studies use data where gasoline prices have been averaged to a weekly level. This granularity is not fine enough to capture the underlying mechanism behind large jumps in prices (Eckert et al. 2004). In an attempt to capture the effects of these mechanisms, the data used in this study contains the average daily gasoline price at each station.

Past studies have attributed too much effect to the proximity of the price to wholesale costs. In fact, price restorations have occurred almost always on Tuesdays and Wednesdays, an effect that suggests that demand factors are driving volatility (Eckert et al. 2004). In fitting the models in this study, a set of dummy variables will inform the model on which day of the week the price occurred.

Using the gained knowledge from this literature, this study will include variables such as information about the day of the week, information about the brand of the gasoline, the proximity and density of other stations and their respective prices, the proximity of major roadways, and the wholesale cost of oil and gasoline on each day in the sample in order to fit a model that can attempt to parse these patterns and make accurate predictions.

3. Data

The raw data set used for this study contains the average daily posted price for gasoline across 13,687 stations in Germany. This represents about 95% of all stations in the country. The data has been collected for the time period of May 16, 2014 to

December 11, 2015 and was provided by Colin Vance, Ph.D., Deputy Director, Environment & Resources of the RWI – Leibniz Institute for Economic Research.

The data set includes the following static variables: brand of station, brand of gasoline, distance to autobahn, whether the operator of the station is independent, the latitude and longitude of the station, and a dichotomous variable indicating if the station lies directly on the autobahn. Also collected every day of the study is the price of E5 gasoline, the price of refined gasoline out of Rotterdam, the price of West Texas Instrument (WTI) crude oil, the price of Brent crude oil, and the day of the week. In total there are 8,270,468 observations in this data set. Each observation is a daily price point for the above mentioned gasoline for each station on each day in the data set, along with the corresponding market and geographic variables for each day and station.

While there are a total of 13,687 stations that are observed on at least one day in the study, only 12,374 of the stations have been observed on all 575 days. In order to keep the task of prediction balanced for each station, only the 12,374 stations that have an observation for each day from May 16, 2014 to December 11, 2015 have been retained for this paper. In total, this means that there are 7,115,050 observations to be used for the purpose of training and testing models to predict the price of gasoline at a specific station.

In order to add to the geographic understanding of the model, reverse geocoding using the latitude and longitude of each station has been performed and variables have been added to the dataset that indicate which of the sixteen German states each station resides in.

The locations of the 12,374 stations that were used for this study can be seen in figure 1 below:

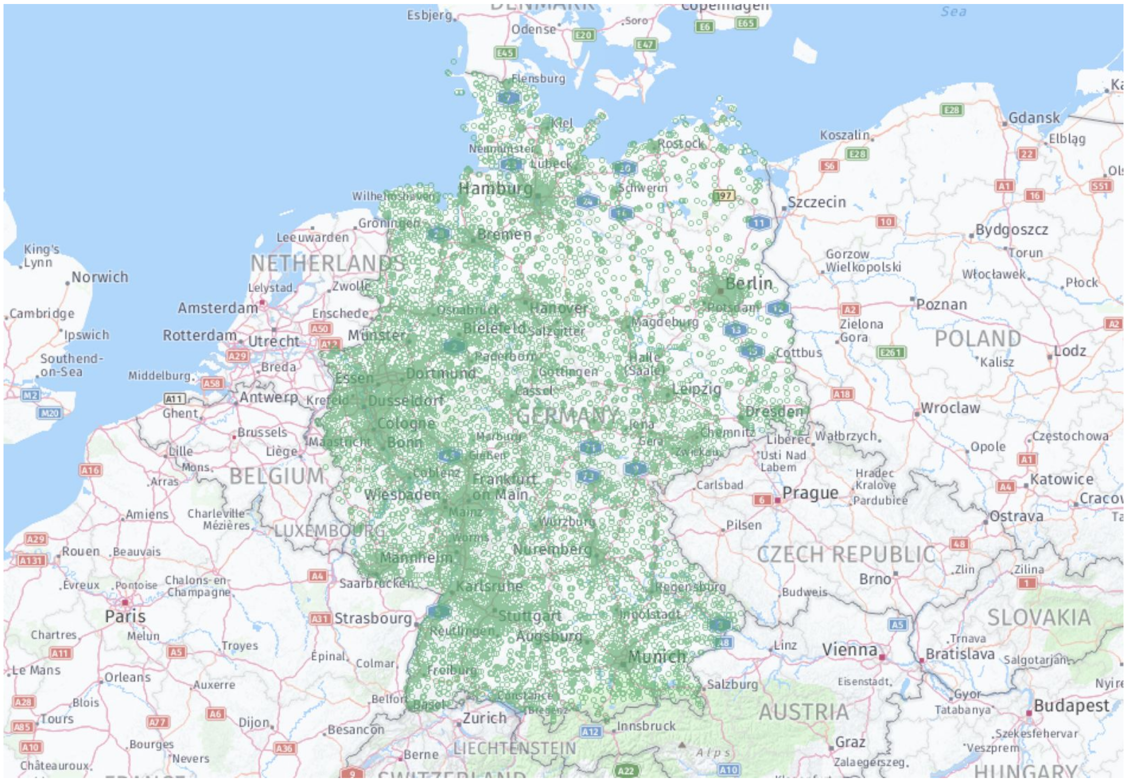


Figure 1: Locations of 12,374 stations used in this study based on latitude and longitude.

Table 1 below shows the descriptive statistics for select variables.

	Observations	Mean	Standard deviation	Minimum	Maximum
E5 gasoline price (€)	7,115,050	1.45364	0.1044921	0.8995	2.498042
Distance to the autobahn (m)	7,115,050	6850.198	7997.5	0.1899418	64109.73
Whether station on autobahn	7,115,050	0.03159851	0.1749287	0	1
Whether station brand is Aral	7,115,050	0.1677791	0.3736701	0	1

Whether station brand is Esso	7,115,050	0.07541521	0.2640602	0	1
Whether station brand is Jet	7,115,050	0.04323118	0.2033771	0	1
Whether station brand is Shell	7,115,050	0.1307829	0.3371628	0	1
Whether station brand is Total	7,115,050	0.02763396	0.1639217	0	1
Rotterdam price ($)	7,115,050	0.4941067	0.0999969	0.321701	0.6813258
Brent crude price ($)	7,115,050	70.62451	22.64122	39.28281	114.6798
WTI crude price ($)	7,115,050	65.40877	21.88225	37.26793	106.68

Table 1: Descriptive statistics for variables in the dataset.

The variable that indicates the day of the week of the observation can take the values 1-7. A value of 1 means that the observation was collected on a Monday. Each incremental value is successive throughout the days of the week. For the purposes of this study, this variable was one-hot encoded so that six dummy variables are created for each day other than Sunday, the baseline day.

The variable that indicates which state the station resides in has no missing values and is a categorical variable where each observation contains one of the possible 16 states in Germany. This variable was also be one-hot encoded, resulting in 15 dummy variables indicating whether the station resides in the respective state, with Baden-Wuerttemberg as the baseline state.

This dataset deviates from a common multivariate time series by being multi-site. There are 12,374 sites in the dataset.

A commonly necessary step in time series data analysis is that of removing the time trend and seasonality. A time trend in time series data has been formally defined as “an intrinsically determined monotonic function within a certain temporal span” (Wu, 2007). Neither such a time trend or seasonality trend were found in this time series. Given that the data only spans roughly 1.5 years, time trends that may have been apparent over greater temporal space were not found and no differencing of the data in order to enforce stationarity was performed. The average gasoline price across all stations every day for each day of the time series is shown below.

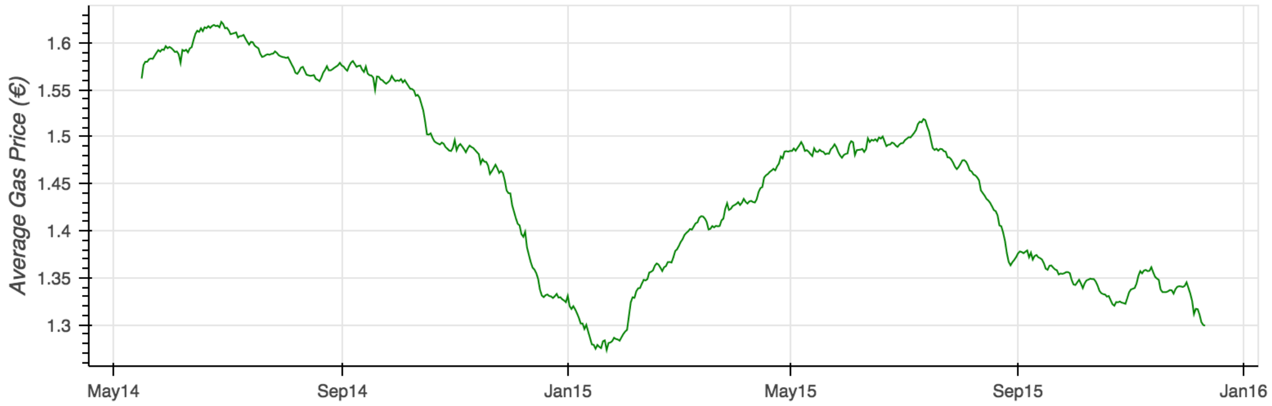


Figure 2: The average daily gasoline price from May 16, 2014 to Dec 11, 2015 across all stations.

4. Methods

There are many appropriate techniques for time series forecasting, such as vector autoregression (Baghestani 2015), moving averages (Baghestani 2015), random forests (Fischer et al. 2017), and LSTM networks (Fischer et al. 2017). In this study, three methods will be compared that can be used to make time series predictions. It is especially enlightening to compare traditional econometric models with more modern machine learning approaches to time series forecasting. For this reason, this paper examines the prediction accuracy of both simple and complex, traditional and modern models. The models considered become increasingly more computationally expensive in regards to both time and computer memory. First, a simple multiple linear regression was considered. Although linear regression is not typically appropriate for time series analysis because of reasons such as the errors being correlated with one another, it is a computationally inexpensive model that can produce unbiased results given stationary data (Shrestha et al. 2018). Second, a random forest, less restrictive in its assumptions than the linear regression, was examined. Third, a long short-term memory network, the neural network most often used for time series prediction, was evaluated.

4.1 Multiple linear regression

While using a multiple linear regression model is a very simplistic approach of modeling this prediction problem, the linear nature of the model may fit well for this problem given the linear relationship between the gasoline price and the refined gasoline price out of Rotterdam. Figure 3 below shows the linear relationship between gasoline price in time t and rotterdam gasoline price in time t averaged across all stations.

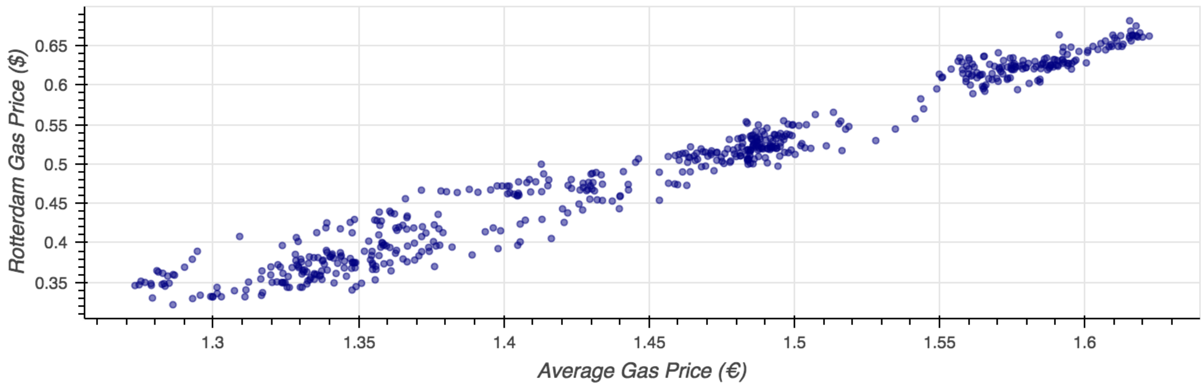


Figure 3: The scatter plot above shows the strongly linear relationship between the average daily gasoline price across all stations and the refined gasoline price out of Rotterdam.

The generalized model for multiple linear regression is as follows:

$$Y = \beta_0 + \beta^T X + \varepsilon$$

(1)

In formula (1), β is an m-dimensional column vector, where m is the number of predictors in X, and X is an m x n dimensional matrix where n is the number of observations in the training set (Kutner et al. 2004).

For this specific application, the model is constructed as below:

$$gas\_price_t = \beta_0 + \beta_1 dautobahn_i + \beta_2 autobahn_i + \beta_3 aral_i + \beta_4 esso_i + \beta_5 jet_i + \beta_6 shell_i + \beta_7 total_i + \beta_8 rotterdam_t + \beta_9 brent_t + \beta_{10} wti_t + \beta_{11} longitude_i + \beta_{12} latitude_i + \beta_{13} num\_days_t + \varepsilon_{it}$$

(2)

The method of ordinary least squares (OLS) was used to fit this linear model. The objective of OLS is to find values for the parameters β so that the squared distance between the true value of the dependent variable and the value predicted by the input variables and β coefficients is minimized (Kutner et al. 2004).

Formula 2 also reveals the data structure for this model. The dependent variable is the gasoline price at the station in time t and the input variables are all of the explanatory exogenous variables appropriate for the station and time period. Dummy variables for each of the weekdays were also added to the model with the baseline day of the week being Sunday. Another set of dummy variables were added to signify which state the station resides in. There are 16 states in Germany and 15 state dummy variables were created, with the baseline state being Baden-Wuerttemberg.

4.2 Random forest

A model that is not held back by the strong linear assumptions of the multiple linear regression is considered next. Random forest regression is the result of multiple decision trees. As described by Kane in a paper comparing the effectiveness of the random forest and other methods for time series forecasting: “Decision trees recursively partition data in the regression space until the amount of variation in the subspace is small. A predictor for the subspace can then be created simply by taking the average value of the dependent data corresponding to the independent data in the subspace...Predictions for new data are obtained by finding the predictor corresponding the partition where the new input variable resides” (Kane 2014).

There are multiple hyperparameters that can be tuned in training in order to help the random forest make the best fit possible. These hyperparameters include the number of trees in the forest, the maximum depth to which each tree is allowed to grow, the minimum number of samples required to split a node, and the minimum number of samples required at each leaf node of a tree (Hastie et al. 2001).

The number of trees in the random forest controls its variability. As more trees are added, the forest's predictions become more stable and the accuracy of the predictions increases up to a point. However, it should be noted that each added tree increases the training time and memory required for the model (Hastie et al. 2001).

Another hyperparameter controls the depth to which each individual tree may grow. Tuning this hyperparameter is an exercise in the balance between underfitting and overfitting. With too small a value for maximum depth, the regression trees will not be able to capture the complexity of the data generating process. However, if the tree is allowed to grow too deep, it will overfit the patterns that are found in the training data – patterns which may be idiosyncratic to the training data and not found in the test sample or the population data. Individual trees are also quite noisy but, when grown appropriately deep, have low bias (Hastie et al. 2001). Once again, it should be noted, that increasing depth adds to training time and memory requirements.

The minimum number of samples required to split a node and the minimum number of samples required at each leaf node of a tree are parameters that help to control overfitting. Regression trees are made up of nodes, a point in the tree where a split based on the value of a feature can be made, and leafs, a value of the dependent variable predicted given the values of the features that led from the root of the tree to that leaf. If nodes are able to be split on too few observations or if leaves can contain too few observations then overfitting may occur since patterns present in only a few observations will affect the the prediction output by the tree (Hastie et al. 2001).

After the random forest has been trained, values for the features at future points in time are passed into the forest and each tree votes on the value that it has predicted for the output variable. Each tree's vote of the value of the continuous variable (in this case, the price of gasoline at each station in time *t*) is averaged and that average value is the prediction emitted by the random forest (Kane 2014).

Regression trees predict a continuous dependent variable, in this case price. However, a regression tree is a piecewise-constant model and it cannot perfectly model smooth regression surfaces with its jagged response. The regression tree will attempt to fit the smooth regression surface of the continuous variable by incorporating enough leaves. This is an important factor in considering which model fits the data best as the random forest is constrained by its jagged response (Hastie et al. 2001).

4.3 Long short-term memory network

The final method to be considered for this particular multisite time series forecast is a long short-term memory network (LSTM). An LSTM is a specialized type of neural network. At its simplest, a neural network has an input layer, a hidden layer and an output layer. All of the data is passed into the model and first goes through the input layer. The input layer is made up of neurons, generally the same number of neurons as the number of features in the dataset. The data exits the input layer and is multiplied by a learned weight that is unique to each edge connecting an input neuron to a hidden layer neuron. This weight is a reflection of the strength of the signal between the two neurons. If the signal is strong enough, the receiving neuron in the hidden layer will use some transformative function to shift the data to a higher level representation in order to fit the relationship between the independent feature data and the dependent variable data that the neural network uses in training. The hidden layer will pass this transformed data matrix to the output layer, where it will again be multiplied by a signal strength-measuring weight on each edge. The output layer uses an activation function in order to transform the output from the hidden layer into the scale of the dependent variable (Bishop 2006).

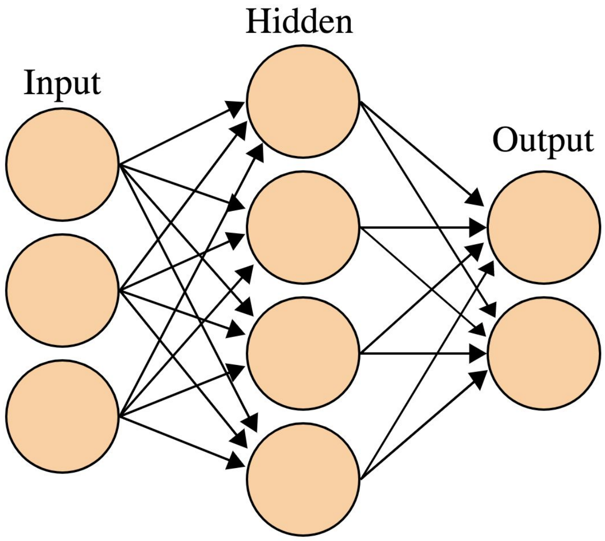


Figure 4: Diagram of a simple neural network (Olah 2015).

However, traditional neural networks are not capable of understanding the type of relationship between the observations in a time series dataset. That is to say that traditional networks are not informed by the output they generate from previous observations. In a time-series problem, the ability to understand that the output from the previous steps should be considered in the following estimation of the output variable is very important. Recurrent neural networks (RNNs) solve this problem by having loops, allowing the outputs from time step *t-1* to be used when estimating the output in time *t* (Fischer et al. 2017).

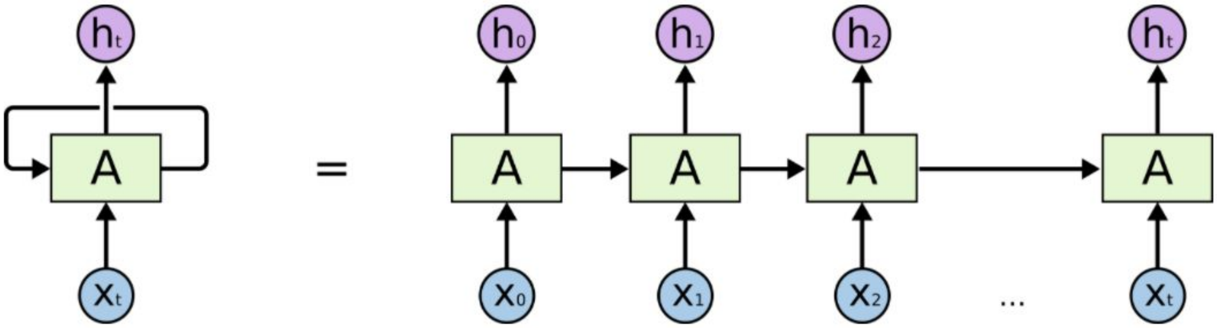


Figure 5: An unrolled recurrent neural network. “In the above diagram, a chunk of neural network, A, looks at some input x_t and outputs a value h_t. A loop allows information to be passed from one step of the network to the next” (Olah 2015).

The problem that persists even with an RNN is that it is possible that important information for forecasting the current observation occurred many observations previously. When the gap between the current observation and the needed information from previous observations grows sufficiently large, the RNN is incapable of collecting that information. LSTM networks solve this problem by being able to learn long-term dependencies. LSTM networks have a hidden layer that is made up of a forget gate, an input gate, and an output gate. The forget gate is shown in the bottom left hand corner of the green hidden layer (A) for X_t in figure 6 below. This gate is made up of a sigmoid layer (σ) which emits a value between 0 and 1 to signify to what degree h_{t-1} and x_t should be remembered and used to influence the output of this layer, h_t. The input gate, made up of the second sigmoid layer (σ) and the tanh layer, will decide which information to update and then replace the information that was forgotten by the forget gate to use when making the prediction h_t. Lastly, the output gate, the final sigmoid layer (σ) and the pink tanh layer, will prepare the information to be passed to the output layer to make the prediction h_t. The sigmoid layer in this gate will collect all of the pertinent information that was kept and updated by the previous two gates and then the tanh layer will transform the values between -1 and 1. These values will then be passed to the output layer where the activation function will scale it to the range of the dependent variable and make the final prediction h_t for the input x_t (Fischer et al. 2017).

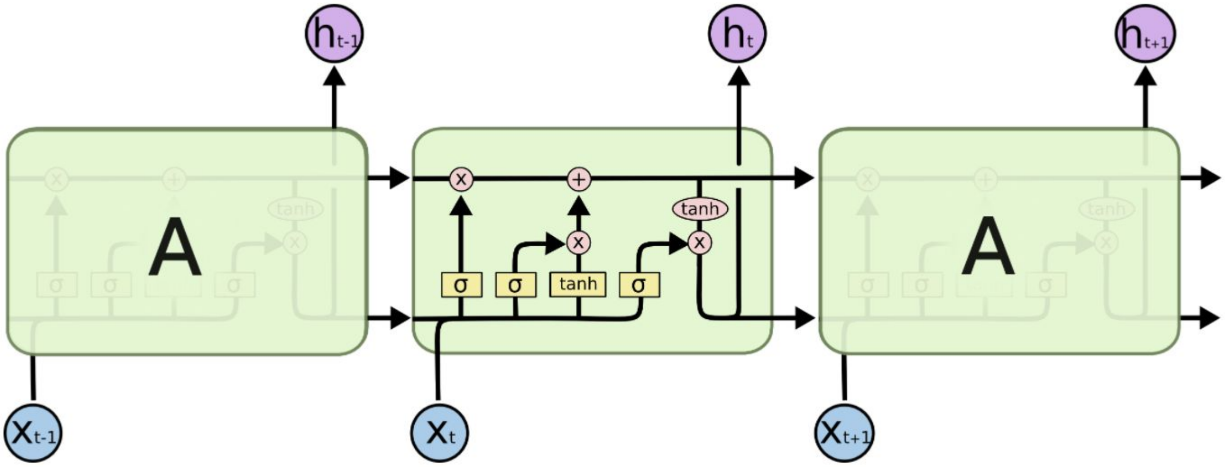


Figure 6: The hidden layer of an LSTM network (Olah 2015).

For the purposes of this study, the data has been formatted into a supervised learning problem to be passed into the LSTM network in the same way that it was organized for both the random forest and the multiple linear regression models. The target variable is the gasoline price at a particular station at time t and the features are the station and day specific variables for time *t*.

The LSTM was optimized using the Adam algorithm. Adam is an alternative to classical mini-batch stochastic gradient descent. It differs by setting and updating a separate learning rate for each parameter in the model using the second moments of the gradients rather than the single learning rate used in stochastic gradient descent (Kingma et al. 2015). The Adam algorithm was chosen because of its numerous benefits, such as being computationally efficient, having small memory requirements, being appropriate for data that is large in terms of size or parameters, being well-suited for “non-stationary objectives”, and having hyperparameters that have “intuitive interpretations and typically require little tuning” (Kingma et al. 2015).

The hyperparameters that were trained for the LSTM network in this study are the batch size, the number of epochs, and whether early stopping was implemented. Mini-batch stochastic gradient descent entails that the neural network iteratively update the weights in the network after each batch of observations has been predicted and compared against the true values of the target variable. The batch size is the number of observations in the training set that the neural network will evaluate and test the correctness of before updating the weights. The weights in the neural network will be updated after each batch of observations so that the observations in the batch are more accurately predicted. This continues until all observations have been evaluated, batch by batch. One entire iteration through the training set is called an epoch. Training the neural network usually requires multiple epochs. In practice, the neural network is trained by experimenting with batch size and number of epochs by comparing the error of the testing set (the set of data that the neural network is not being trained on) after each epoch. A neural network can begin to overfit much like a tree in a random forest. This is apparent once the error on the testing set begins to increase even as the error on the training set continues to decrease. The technique used in this study to stop overfitting is known as early stopping. By keeping track of the error from the testing set after each epoch, a rise in the testing error triggers the training to stop, regardless of the number of epochs set at the beginning of training. This halts overfitting by not allowing the model to continue updating the weights in a way that too closely fits the patterns found in the training set (Bengio 2012). In the implementation of the LSTM network, the Python deep learning package Keras was used. More discussion of the software can be found in the appendix. Keras includes a parameter for early stopping called patience. Patience is the number of epochs after which training will be stopped if no improvement in the testing error is observed (Chollet 2015).

Before training the LSTM network, the variables were standardized by converting each to the range of 0 to 1. This standardization of all input variables is appropriate for the neural network due to the nature of the sigmoid function used in optimization. “Without this standardization, large values input into a neural network would require extremely small weighting factors to be applied” (Dawson, 1998). This can cause multiple problems, such as inaccurate floating point calculations and the fact that changes made during optimization “would be insignificantly small, and training would be very sluggish, as the gradient of the sigmoid function at extreme values would be approximately zero" (Dawson, 1998). All predictions were made using the standardized input variables and the predicted gasoline prices were transformed back into the original scale so that the results from each of the models can be compared on the same scale.

5. Data Preparation

The linear model, random forest, and LSTM network all rely on the same inputs for the prediction that were used to train or fit the models. Three of these inputs, WTI, Brent crude oil, and Rotterdam gasoline prices, are variables whose future values cannot be known when predictions are made. For this reason, these three variables have been treated separately as univariate time series. Simple linear models have been fit using the values of the initial 545 days in the time series in order to predict the values of the final 30 days in each time series. When predictions were made for the final 30 days of gasoline prices at each of the 12,374 stations, the predicted values (rather than the observed values for these "future" days) were passed into each model as the values for the variables WTI, Brent crude oil, and Rotterdam gasoline price.

The training set is comprised of the first 545 days of data, roughly 95% of the available data, and the remaining 30 days, roughly 5% of the data, is used as the testing set.

6. Results

All analysis was done using the programming language Python. The StatsModel package was used for the multiple linear regression (Seabold 2009). For the random forest, the scikit-learn package was used (Pedregosa 2001). The Keras package was used (Chollet 2015) for the LSTM network. More information about these packages and the classes used can be found in the appendix.

6.1 Persistence model

In measuring the overall prediction quality of the models, it is important to begin with a baseline of error for the predictions. It is customary to use the most naive model possible in order to create these baseline error measures. A persistence model is one such extremely naive approach for a time series prediction problem. With a persistence model, the previously observed values for the target variable will persist through time and are treated as the predicted values in the future (Lei et al. 2009).

In this application, the values that are being predicted are the gasoline prices at each station for the final month of the data set, November 12, 2015 to December 11, 2015. By using the observed gasoline prices in the final month of the training portion of the dataset, the entirety of which runs from May 11, 2014 to November 11, 2015, I employ a persistence model which naively predicts that the prices from the last month of the training set will be the prices for the month I set out to predict.

With these predictions for the gasoline price at each station for the month of November 12, 2015 to December 11, 2015, I calculate the root mean squared error

(RMSE) of these predictions compared to the actual observed prices for that month. The result is a RMSE of 0.034. Since the RMSE has the same units as the quantity being estimated, the persistence model can predict gasoline prices one month in the future that are on average deviant from the true values by 3.4 cents. This will be used as our baseline when evaluating the results of the models created for this study since it is a very naive and computationally inexpensive process by which to make predictions.

6.2 Multiple linear model

The linear model was fit using the data from the first 545 days of the dataset. With its simple formulation and computationally inexpensive training method, the linear model requires the least amount of training time and computation power out of the three non-baseline models. This aspect makes the linear model an attractive option for this multi-site prediction problem, where training deeper, more complex models can be very temporally and computationally expensive. However, the predictions from this model were not accurate enough to make this simple solution viable. The gasoline prices predicted by the linear model averaged over all stations and days for the final 30 days had an RMSE of 0.035, a higher RMSE than the one produced by the baseline persistence model.

Figure 7 below shows the average predicted gasoline price compared to the average observed gasoline price over all stations on each of the 30 days in the prediction time period. It can be seen that though the linear model mirrored the general trend of the gasoline prices each day, there are substantial deviations from the true average daily prices.

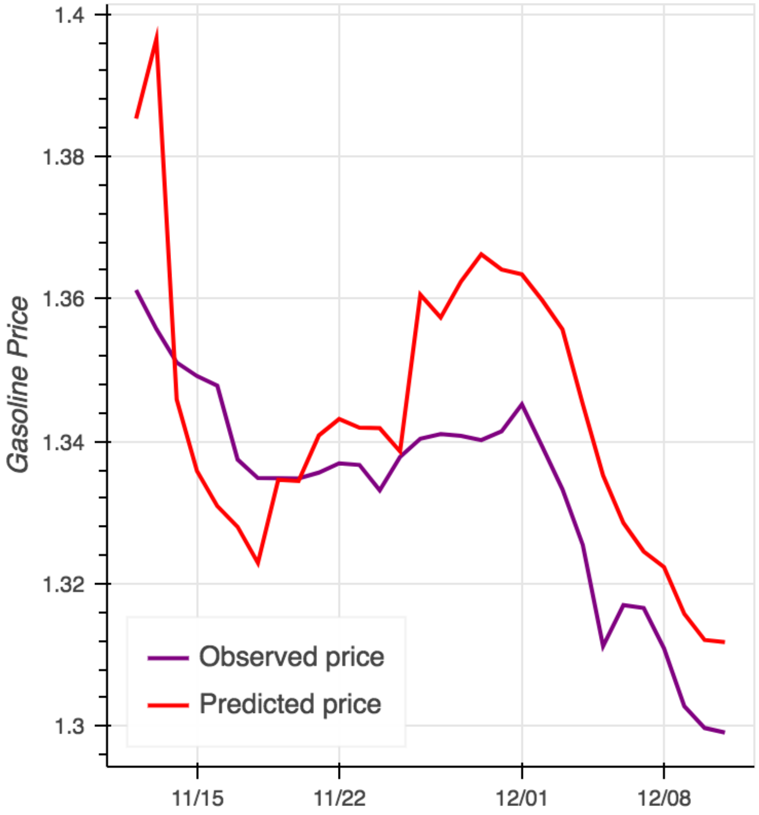


Figure 7: The observed and predicted gasoline prices averaged across all stations each day from November 12, 2015 to December 11, 2015.

While the computational ease of training and testing the multiple linear model is advantageous, in its simplest form the multiple linear model is not accurate enough to beat out the baseline persistence model. It should be noted that the persistence model is computationally less expensive than even the linear model, since it only requires shifting values in the dataframe.

6.3 Random forest

The random forest model was trained on the same initial 545 days of data that was used to fit the linear model. The same predictions for the non-static WTI, Brent crude oil, and Rotterdam gasoline prices were used in place of the observed values during prediction. However, the random forest model has tuning hyperparameters that go beyond the simplicity of the linear model and allow the model to be better specified for this exercise in prediction. As discussed previously, the hyperparameters for the random forest model that were tuned are the number of trees in the forest, the maximum depth to which each tree is allowed to grow, the minimum number of samples required to split a node, and the minimum number of samples required at each leaf node of a tree.

I performed a random grid search across different values of each of these parameters. Most notably, I trained the random forest models with between 75 and 170 trees and with maximum depths of between 15 and 100. With each added tree and increase in maximum depth, the model takes longer to train and uses more memory.

Several combinations of hyperparameters yielded the same lowest RMSE. The most computationally efficient combination was chosen as the final model. This model was trained with 150 trees, a maximum depth of 80, a minimum number of samples required to split a node of 2, and a minimum number of samples required at each leaf node of 1. The RMSE for this model was 0.0267.

The RMSE for the random forest model is a 0.83 cent improvement in accuracy over the multiple linear model and a 0.73 cent improvement over the baseline.

Figure 8 shows the average predicted gasoline price across all stations on each of the 30 days that were predicted in addition to the average observed gasoline price on each of the 30 days. As was discussed previously, the random forest is a piecewise-constant model and it will often have a very jagged prediction pattern when attempting to fit a smooth regression line, as compared to classification problems at which it excels (Hastie et al. 2001). For this reason, figure 8 reveals that the random forest has predicted very irregular gasoline prices as compared to the observed prices. However, the prices predicted by the random forest were on average more correct than those predicted by the baseline or the multiple linear model.

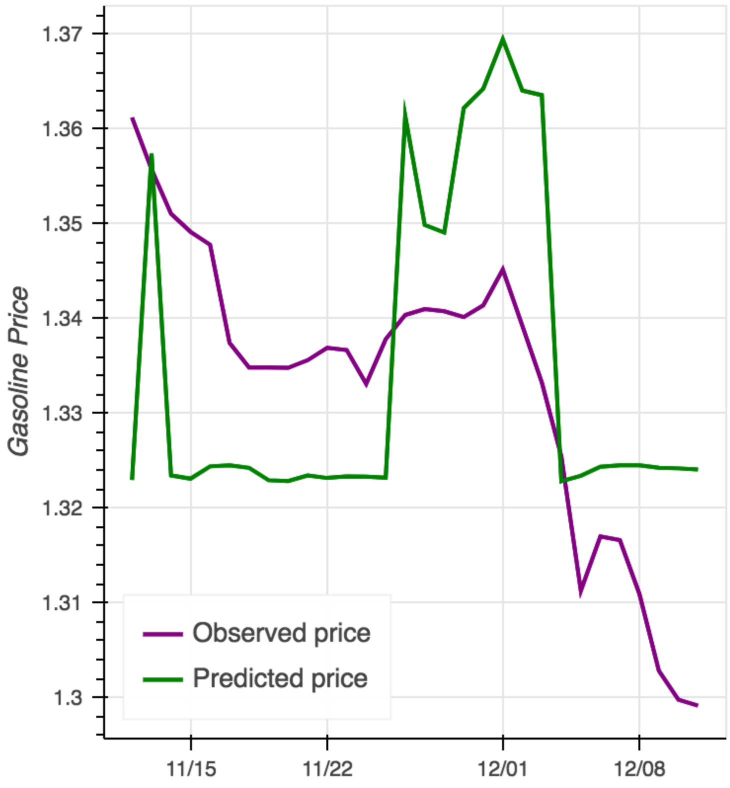


Figure 8: This figure shows both the predicted and observed gasoline prices, averaged across all stations, for each day of the 30 day prediction interval that resulted from the tuned random forest model.

6.4 Long short-term network

The neural network model that was created for this study is the originally developed LSTM comprised of one hidden LSTM layer and a feedforward output layer (Hochreiter et al. 1997). The model was trained using the same input data of the initial 545 days that were used to train the linear and random forest models.

I trained the LSTM on a series of values for the number of epochs and batch size hyperparameters. The number of epochs tested ranged from 10 to 500 and batch sizes ranged from 500 to 7200.

Training the LSTM model requires significant time; in fact, training the 500 epoch model required more time than any other model trained for this study. The model trained using 500 epochs and a batch size of 1000 took nearly 10 hours to train using high memory CPUs on Google Cloud Compute Engine. However, the LSTM model trained with 500 epochs required less memory than training the random forest with the most trees and depth. In order to improve the time intensive aspect of training the LSTM model and, more vitally, to control for overfitting, early stopping was utilized.

While tuning the LSTM model, I employed the early stopping technique with varying patience periods for combinations of batch size and number of epochs where it was apparent that overfitting was occurring.

The combination of hyperparameters that yielded the lowest RMSE was chosen. With the number of epochs set to 20 and batch size set to 1000, the RMSE of the LSTM model is 0.0298. The relationship between train and test loss is shown in figure 9 below.

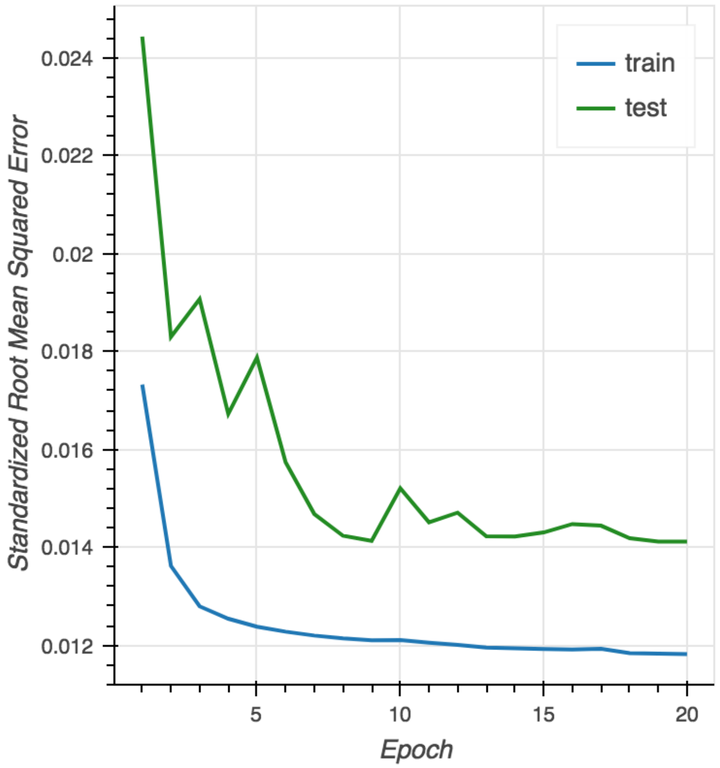


Figure 9: The test loss and train loss are graphed at each epoch as the LSTM model is trained.

Figure 10 shows the predicted and the observed gasoline prices each day of the 30 days in the prediction period averaged across all stations. This figure shows how well fit the LSTM model was to the general trends of the changes in gasoline prices across the prediction period. However, the LSTM model did not outperform the random forest model in the metric of RMSE averaged across all stations and days.

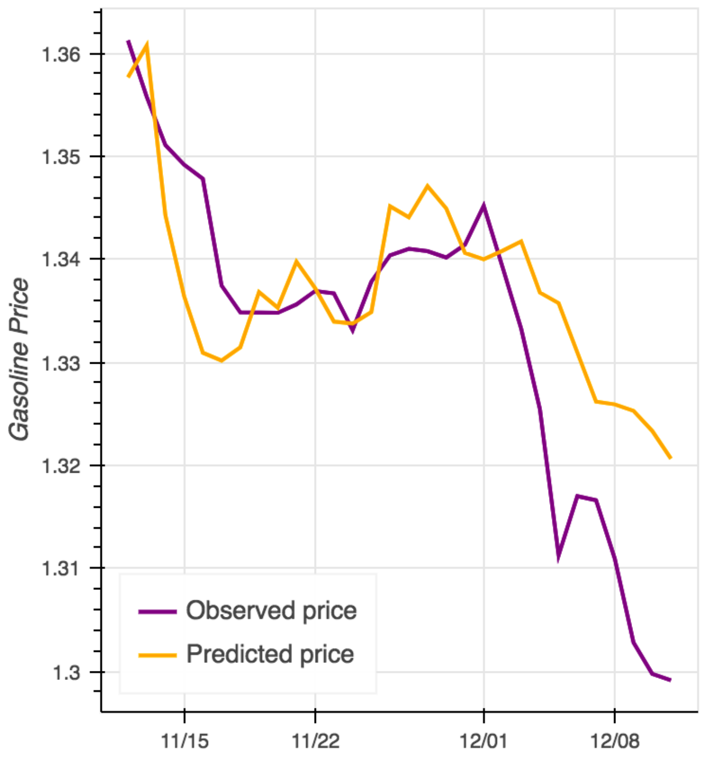


Figure 10: This plot shows the predicted and observed gasoline prices, averaged across all stations, for each of the 30 days in the prediction interval predicted by the tuned LSTM model.

6.5 Comparing the Methods

If only the RMSE averaged across all days and all stations were considered in determining the most appropriate model for this application, the random forest would be considered the best model. The following RMSE values averaged over all stations and days were found: 0.034 cents for the multiple linear model, 0.0298 for the LSTM model, and 0.0267 for the random forest model.

However, when observing the RMSE over each individual day in the prediction period and averaged across all stations, it can be seen that the random forest is not always the most accurate model. Figure 11 below shows that there are multiple days where the LSTM model outperforms the random forest and even three days where the multiple linear regression has a lower RMSE than the random forest.

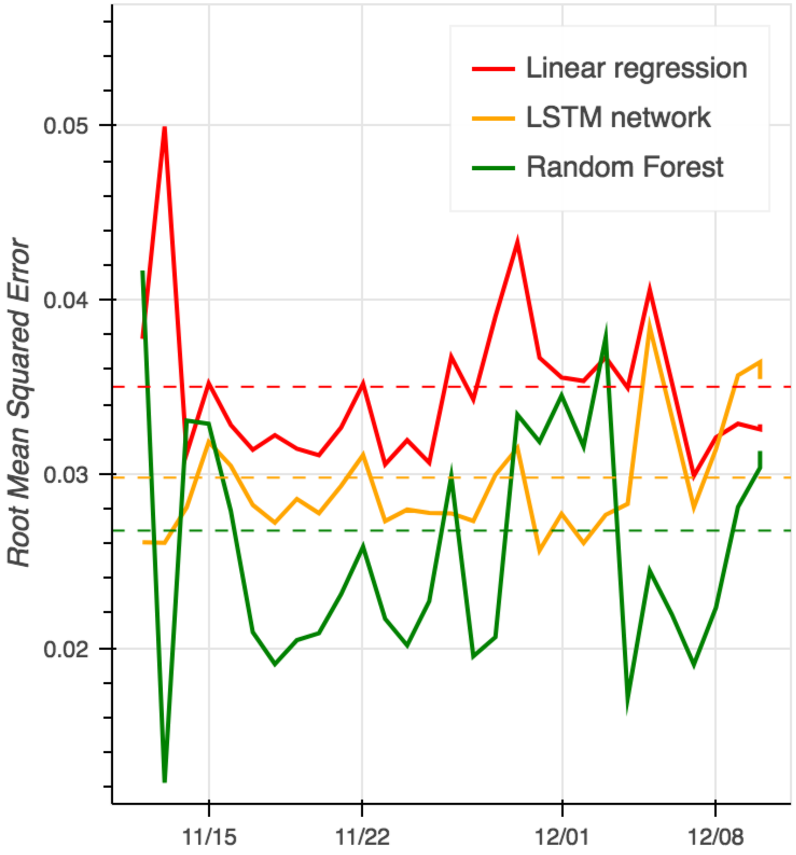


Figure 11: The plot above shows the RMSE averaged over all stations on each day of the prediction period for each model. The average RMSE over all stations and all days is shown by the dashed lines in the appropriate color for each model.

Similarly, it may be informative to look at the RMSE for each model in each state rather than over the country as a whole. Figure 12 below shows the RMSE in each of the 16 states in Germany averaged across all days and stations in each state. By looking at the performance of each model spatially, we can see that the linear model is consistently outperformed by the LSTM network and the random forest model.

However, the LSTM network does perform better than the random forest in one state, Thuringia.

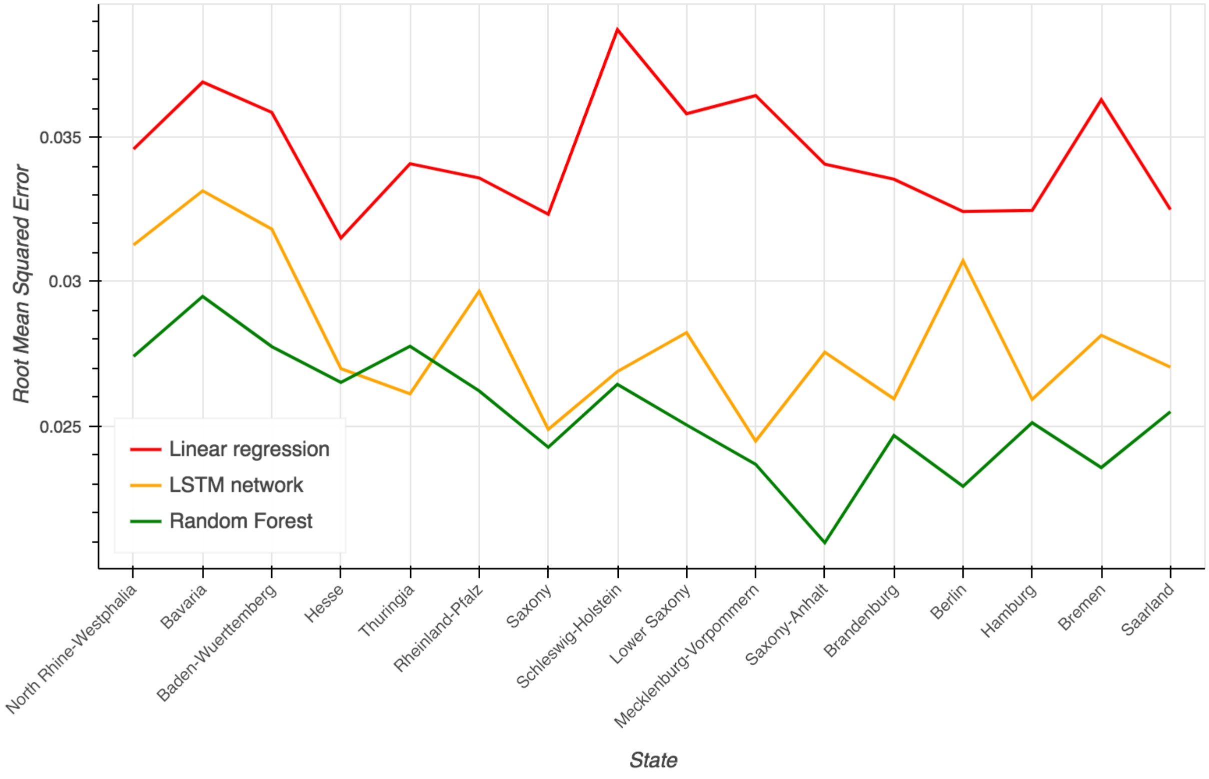


Figure 12: The RMSE for each state by method, averaged across all stations and days.

Measuring the performance of the models using a finer temporal or spatial lens illustrates that the random forest model is not more accurate in all cases. However, it is the most accurate model of the three tested on average and across most days and most states.

7. Conclusion

A multiple linear model, a random forest, and a long short-term memory network were each trained to predict gasoline prices every day at each of the 12,374 stations in the dataset across Germany. The random forest model proved to be the most accurate when averaged across all days in the prediction period and all stations and is a very promising method to use for these types of multisite time series prediction problems.

When using a finer lens than averaging across all days and all stations, the choice of the most accurate model is not as clear. The random forest is not the most accurate on all of the 30 days in the prediction set nor is it always the most accurate when comparing performance at a state level.

Investigations into training these models for finer spatial areas would be a valuable extension to this work. Each of the three tested models shows different spatial patterns in accuracy. Further work should explore which models excel in areas that are urban, suburban, sparsely surrounded by other gasoline stations, bordering another state, country or body of water, etc. The appendix contains a cartogram map for each of the models with the color and size of the circles relating to the RMSE averaged across all days for each individual station.

It would also be valuable to fit a dynamic panel model to the data and compare its accuracy to the three models tested in this study. Similarly to a VAR model, a dynamic panel model addresses the serial correlation present in time series datasets. However, since this dataset contains a large cross-sectional dimension and a comparatively short time series dimension, the assumptions of the VAR model are violated and the dynamic panel model is a more appropriate choice (Bond 2002). Fitting a dynamic panel model would be an improvement over the multiple linear regression and would be an interesting extension in comparing the traditional econometric methods to the machine learning methods used in this study.

Another extension to this work would be to tune a model of stacked LSTM networks. The LSTM network used in this paper was a neural network with a single hidden LSTM layer. However, there is promising work on time series prediction using multiple LSTM hidden layers in a neural network in order to allow the model to learn higher-order temporal patterns (Li et al. 2018).

References

Allcott, Hunt and Nathan Wozny. “Gasoline prices, fuel economy, and the energy paradox”. *Rev. Econ. Stat.*, 96, 5 (2014). 779-795.

Alquist, R., L. Kilian, and R. Vigfusson. “Forecasting the price of oil”. *Handbook of Economic Forecasting*, 2A (2011).

Anderson, S.T., R. Kellogg, J.M. Sallee, and R.T. Curtin. “Forecasting gasoline prices using consumer surveys”. *American Economic Review*, 101, 3 (2011). pp. 110-114.

Bengio Yoshua. “Practical Recommendations for Gradient-Based Training of Deep Architectures”. In *Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science,* edited by Montavon G., Orr G.B., Müller KR, 7700. Berlin, Heidelberg: Springer, 2012.

Bishop, Christopher M. *Pattern Recognition and Machine Learning.* Springer, 2006.

Bond, Stephan R. “Dynamic panel data models: a guide to micro data methods and practice”. *Portuguese Economic Journal* 1, 2 (2002). 141-162. doi.org/10.1007/s10258-002-0009-9.

Borenstein, Severin, A. Colin Cameron, and Richard Gilbert. “Do Gasoline Prices Respond Asymmetrically to Crude Oil Price Changes?” *The Quarterly Journal of Economics* 112, 1 (1997). 305–339. https://doi.org/10.1162/003355397555118.

Borenstein, Severin and Andrea Shepard. "Dynamic Pricing in Retail Gasoline Markets". *National Bureau of Economic Research, Working Paper Series* 4489 (1993). 10.3386/w4489.

Chollet, Francois. “Keras: The Python Deep Learning Library”. Keras Documentation, 2015. https://keras.io.

Dawson, Christian W. and Robert Wilby. “An artificial neural network approach to rainfall-runoff modelling”. *Hydrological Sciences Journal* 43, 1 (2018). 47-66. doi.org/10.1080/02626669809492102

Eckert, Andrew and Douglas S. West. "Retail Gasoline Price Cycles across Spatially Dispersed Gasoline Stations." *The Journal of Law and Economics* 47, 1 (2004). 245-273.

Fischer, Thomas and Christopher Krauss. “Deep learning with long short-term memory networks for financial market predictions”. *European Journal of Operational Research* (2017). doi.org/10.1016/j.ejor.2017.11.054.

Haining, Robert. "Testing a Spatial Interacting-Markets Hypothesis." *The Review of Economics and Statistics* 66, 4 (1984). 576-83. doi:10.2307/1935981.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. New York, NY, USA: Springer New York Inc., 2001.

Haucap, J., U. Heimeshoff, and M. Siekmann. "Fuel Prices and Station Heterogeneity on Retail Gasoline Markets." *Energy Journal* 38, 1 (2018). 81-103. 10.5547/01956574.38.6.jhau.

Hosken, Daniel S., Robert S.McMillan and Christopher T. Taylor. “Retail gasoline pricing: What do we know?” *International Journal of Industrial Organization* 26, 6 (2008). 1425-1436. doi.org/10.1016/j.ijindorg.2008.02.003.

Kane, MJ, Price N, Scotch M, and Rabinowitz P. “Comparison of ARIMA and Random Forest time series models for prediction of avian influenza H5N1 outbreaks.” *BMC Bioinformatics* 15, 1 (2014). doi:10.1186/1471-2105-15-276.

Kingma, Diederik P. and Jimmy Lei Ba. “Adam: A Method for Stochastic Optimization”. *ICLR* (2015). https://arxiv.org/abs/1412.6980.

Kutner, Michael H., Chris Nachtsheim, John Neter, and William Li. *Applied Linear Statistical Models.* 5th ed. McGraw-Hill, 2004.

Kvasnička, M., R. Staněk, and O. Krčál. “Is the Retail Gasoline Market Local or National?” *Journal of Industry, Competition and Trade* 18, 1 (2018). 47-58.

Lei, Ma, Luan Shiyan, Jiang Chuanwen, Liu Hongling, Zhang Yan. “A review on the forecasting of wind speed and generated power”. *Renewable and Sustainable Energy Reviews* 13, 4 (2009). doi.org/10.1016/j.rser.2008.02.002.

Li, YiFei and Han Cao. “Prediction for Tourism Flow based on LSTM Neural Network”. *Procedia Computer Science* 129 (2018). 277-283. doi-org.proxy.uchicago.edu/10.1016/j.procs.2018.03.076.

Lin, Lin, Fang Wang, Xiaolong Xie, and Shisheng Zhong. “Random forests-based extreme learning machine ensemble for multi-regime time series prediction”. *Expert Systems with Applications* 83 (2017). 164-176. doi.org/10.1016/j.eswa.2017.04.013.

Medsker, L.R. and L.C. Jain. *Recurrent neural networks: Design and applications International series on computational intelligence*. CRC-Press, 2000.

Olah, Christopher. “Understanding LSTM Networks.” Understanding LSTM Networks -- Colah's Blog, 27 Aug. 2015, colah.github.io/posts/2015-08-Understanding-LSTMs/.

Pinkse, J. and Margaret E. Slade. “Contracting in space: An application of spatial statistics to discrete-choice models”. *Journal of Econometrics* 85, 1 (1998). 125-154. https://doi.org/10.1016/S0304-4076(97)00097-3.

Pinkse, J. and Margaret E. Slade and C. Brett. “Spatial Price Competition: A Semiparametric Approach.” *Econometrica* 70 (2002). 1111–1153. doi:10.1111/1468-0262.00320.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, E. “Scikit-learn: Machine Learning in Python”. *Journal of Machine Learning Research* 12 (2001). 2825-2830.

Pravilovic, Sonja, Massimo Bilanci, Annalisa Appice and Donato Malerba. “Using multiple time series analysis for geosensor data forecasting”. *Information Sciences* 380 (2017). 31-52. doi.org/10.1016/j.ins.2016.11.001.

Sak, Haşim, Andrew Senior and Françoise Beaufays. “Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition”. *CoRR* 1402.1128 (2014). arxiv.org/abs/1402.1128.

Satoh, E., Iwase, R., Kamakura, K., Sawasato, S., Tominaga, S. “Consumer search costs, geographical concentration, and retail gasoline pricing: Evidence from inland Japan”. *Japan and the World Economy* 45 (2018). 1-8. 10.1016/j.japwor.2017.11.002.

Seabold, Skipper and Josef Perktold. “StatsModel: Statistics in Python”. StatsModel documentation, 2009. http://statsmodels.sourceforge.net/.

Shepard, Andrea. “Contractual Form, Retail Price, and Asset Characteristics in Gasoline Retailing”. *The RAND Journal of Economics* 24, 1 (1993). 58-77. http://www.jstor.org/stable/2555953.

Shrestha, Min B. and Guna R. Bhatta. “Selecting appropriate methodological framework for time series data analysis”. *The Journal of Finance and Data Science* (2018). doi.org/10.1016/j.jfds.2017.11.001.

Tay, Francis E.H. and Lijuan Cao. “Application of support vector machines in financial time series forecasting”. *Omega* 29, 4 (2001). 309-317. doi.org/10.1016/S0305-0483(01)00026-3.

Wu, Zhaohua, Norden E. Huang, Steven R. Long and Chung-Kang Peng. “On the trend, detrending, and variability of nonlinear and nonstationary time series.” *PNAS* 104, 38 (2007). 14889-14894. https://doi.org/10.1073/pnas.0701020104.

Zang, Chuanyun. “Deep Learning in Multiple Multistep Time Series Prediction”. *arXiv preprint* (2017). arXiv:1710.04373.

Appendix

1. Software

1a. Univariate linear prediction of exogenous prices

The software used to make the univariate time series predictions for the WTI crude price, Brent crude price, and Rotterdam gasoline prices in the time period I sought to predict was scikit-learn (Pedregosa 2001). Scikit-learn is an open source software package for the Python programming language that provides models for many applications, such as classification, clustering, and regression. In this case, a simple linear model was used and the data was organized as a supervised learning problem where each observation contained the dependent variable, the price at time *t+1*, and the sole predictor, the price at time *t*.

1b. Multiple linear regression

The multiple linear regression was estimated using StatsModel (Seabold 2009). StatsModel is a module developed for the Python programming language that provides users with classes and functions so that they may perform statistical tests and create statistical models (Seabold 2009). For the multiple linear regression, I fit an ordinary least squares model and then used the model to make predictions of the gasoline prices on future days.

1c. Random forest

The random forest was estimated using scikit-learn (Pedregosa 2001). Scikit-learn is an open source software package for the Python programming language that provides models for many applications, such as classification, clustering, and regression. For this model, the data was split and the training set was used to train an instance of the RandomForestRegressor from scikit-learn.

1d. Long short-term network

The LSTM network was fit using Keras (Chollet 2015). Keras is an open source package for the Python programming language that acts as a “high-level neural networks API”. Essentially, Keras is wrapper around the lower-level deep learning libraries TensorFlow, Theano, and CNTK that allows users to create deep learning models faster than the more labor-intensive libraries that it draws from (Chollet 2015). The analysis done in this study used TensorFlow as the lower-level library. I instantiated a Sequential model and then added an LSTM layer and a Dense layer for the output layer. A Sequential model is a model which processes small batches of the data at a time and then discards it before continuing to process the next batch. Because of this, it is memory efficient and appropriate for large datasets (Bishop 2006). The model was optimized with the Adam algorithm and early stopping was used to monitor the testing loss to determine whether or not to stop training after each epoch.

2. Results

2a. Multiple linear regression

The following is a table showing the detailed results of the multiple linear regression.

	Coefficient	**Standard Error**	**p-value**
Constant	0.8937	0.001	0.000
Whether station is on Autobahn	0.0517	7.05e-05	0.000
Distance to Autobahn	-2.313e-08	1.63e-09	0.000
Whether station brand is Shell	0.0360	3.77e-05	0.000
Whether station brand is Total	0.0275	7.95e-05	0.000
Whether station brand is Esso	0.0127	4.74e-05	0.000
Whether station brand is Aral	0.0382	3.41e-05	0.000
Whether station brand is Jet	-0.0093	6.09e-05	0.000
Rotterdam price ($)	0.8815	0.000	0.000
Brent crude price ($)	-0.0001	3.91e-06	0.000
WTI crude price ($)	0.0013	3.8e-06	0.000
Latitude	0.0002	2.35e-05	0.000
Longitude	-0.0010	1.64e-05	0.000
Number of days since beginning	0.0001	1.88e-07	0.000

Below are the diagnostics for the multiple linear regression model:

No. Observations:	6743830	**R-squared:**	0.906
F-statistic:	1.912e+06	**Jarque-Bera:**	9946243.747
Durbin-Watson	1.415	**P(Jarque-Bera):**	0.00

2b. Random forest

The following tables show the various combinations of hyperparameters that were tested for this data. The memory-intensive nature of each additional decision tree created an upward bound for the number of trees and the depth allowed in this iteration of the study.

Max depth	Number of trees	RMSE
15	100	0.0330
16	100	0.0324
17	100	0.0318
18	75	0.313
25	75	0.0278
25	100	0.0278
30	75	0.0270
30	100	0.0269
40	100	0.0268
50	100	0.0268
60	100	0.0269
100	100	0.0269
70	100	0.0269
70	150	0.0267
80	150	0.0267
70	170	0.0268
100	150	0.0267

2c. Long short-term network

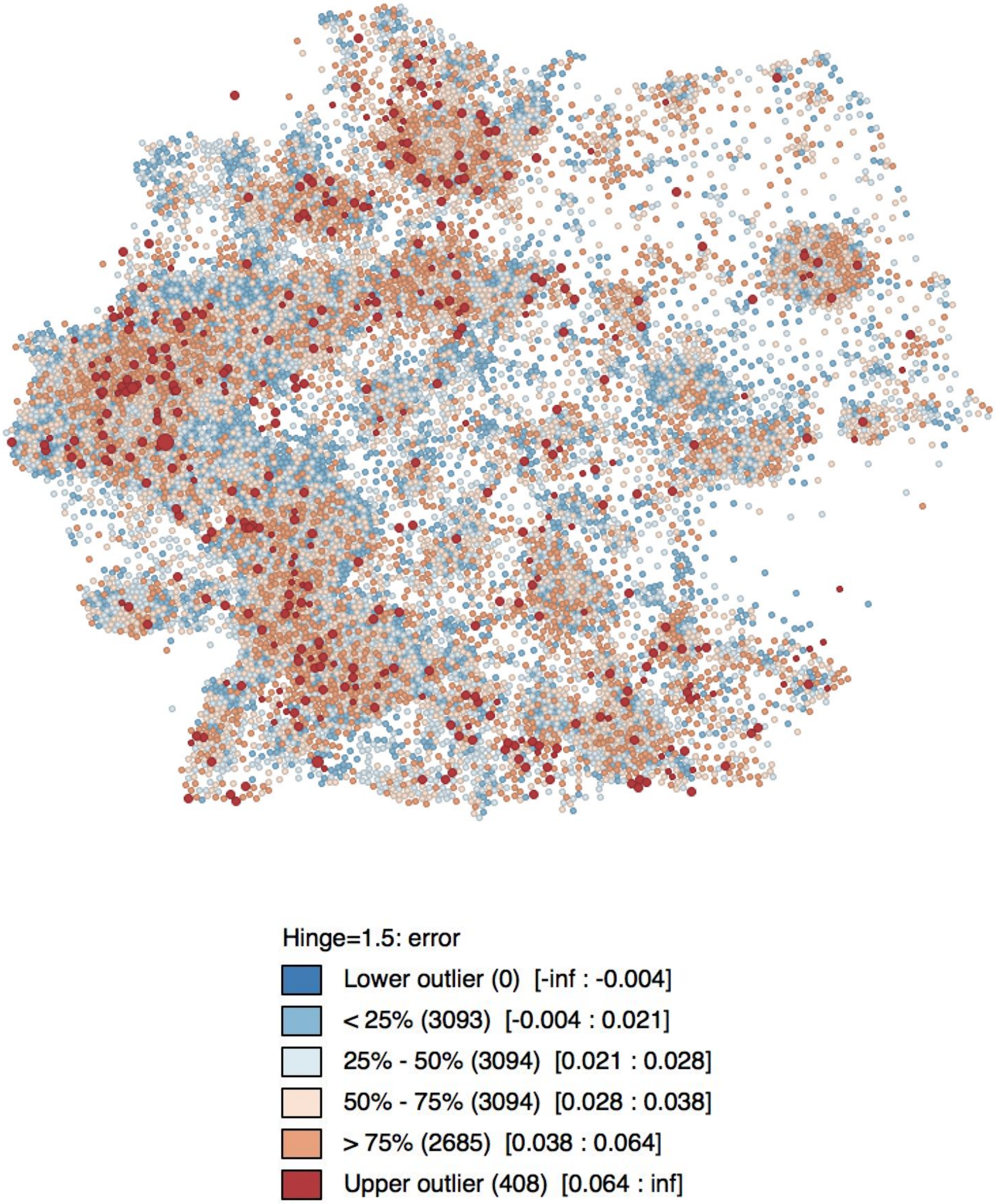
The following table shows the hyperparameters that were altered in training, including the patience parameter for early stopping and the epoch on which the early stopping stipulation ceased training (if applicable).

Number of epochs	Batch size	Early stopping patience	Early stopping epoch	RMSE
10	7200	5	7	0.0321
10	1000	--	--	0.0362
15	1000	--	--	0.0301
20	1000	--	--	0.0298
20	3000	--	--	0.0304
25	1000	5	17	0.0305
30	1500	--	--	0.0305
30	500	--	--	0.0337
30	500	5	12	0.0353
40	1000	--	--	0.0316
50	1000	--	--	0.0300
50	1000	3	13	0.0315
100	1000	5	16	0.0308
500	1000	--	--	0.0385

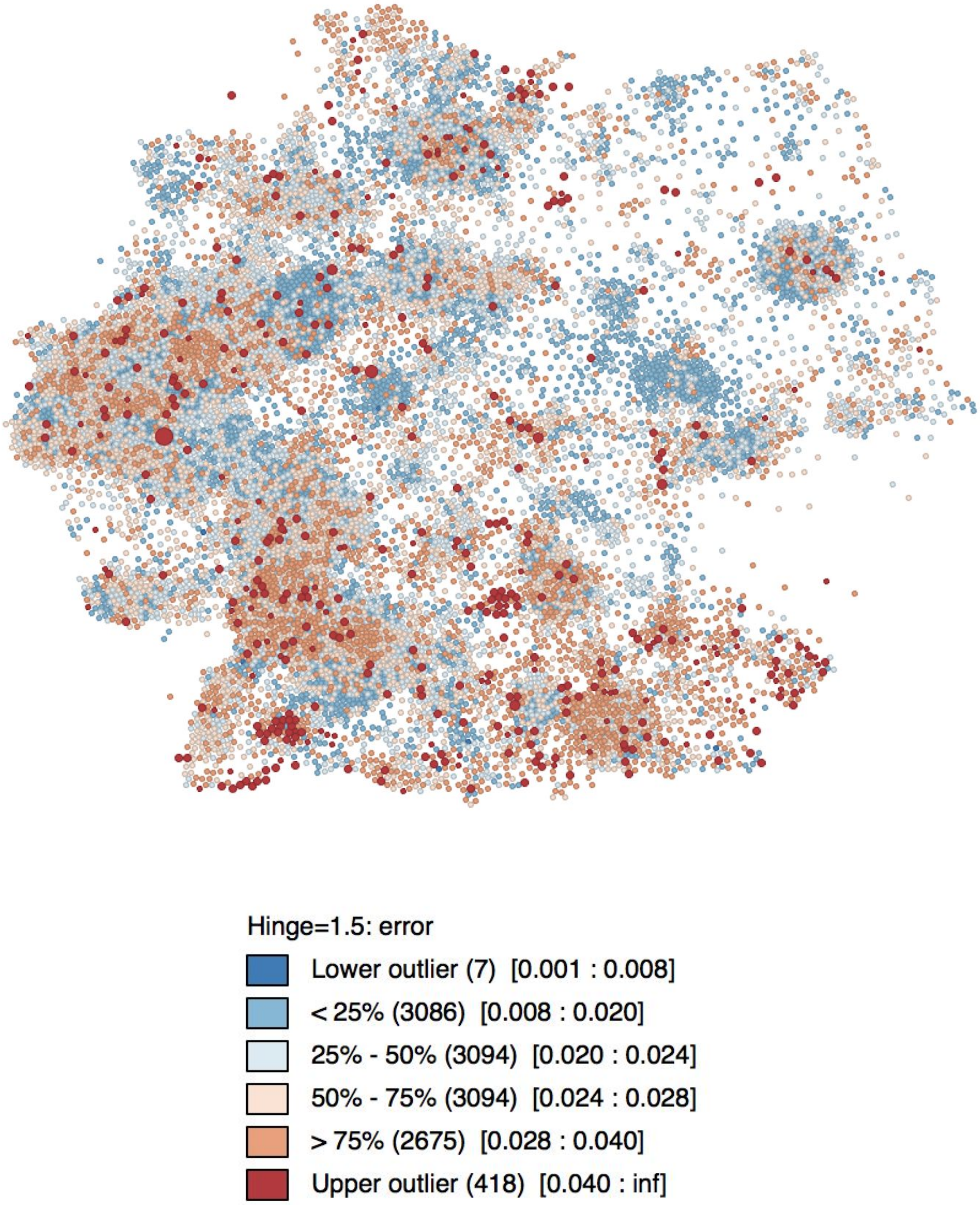
3. Cartogram Maps

Each of the cartogram maps found starting on the next page have a circle for each of the 12,374 stations used for this study. The circle's size and color corresponds to the RMSE between predicted and observed gasoline prices averaged across all 30 days of the prediction period. The legend for the colors is shown below each map.

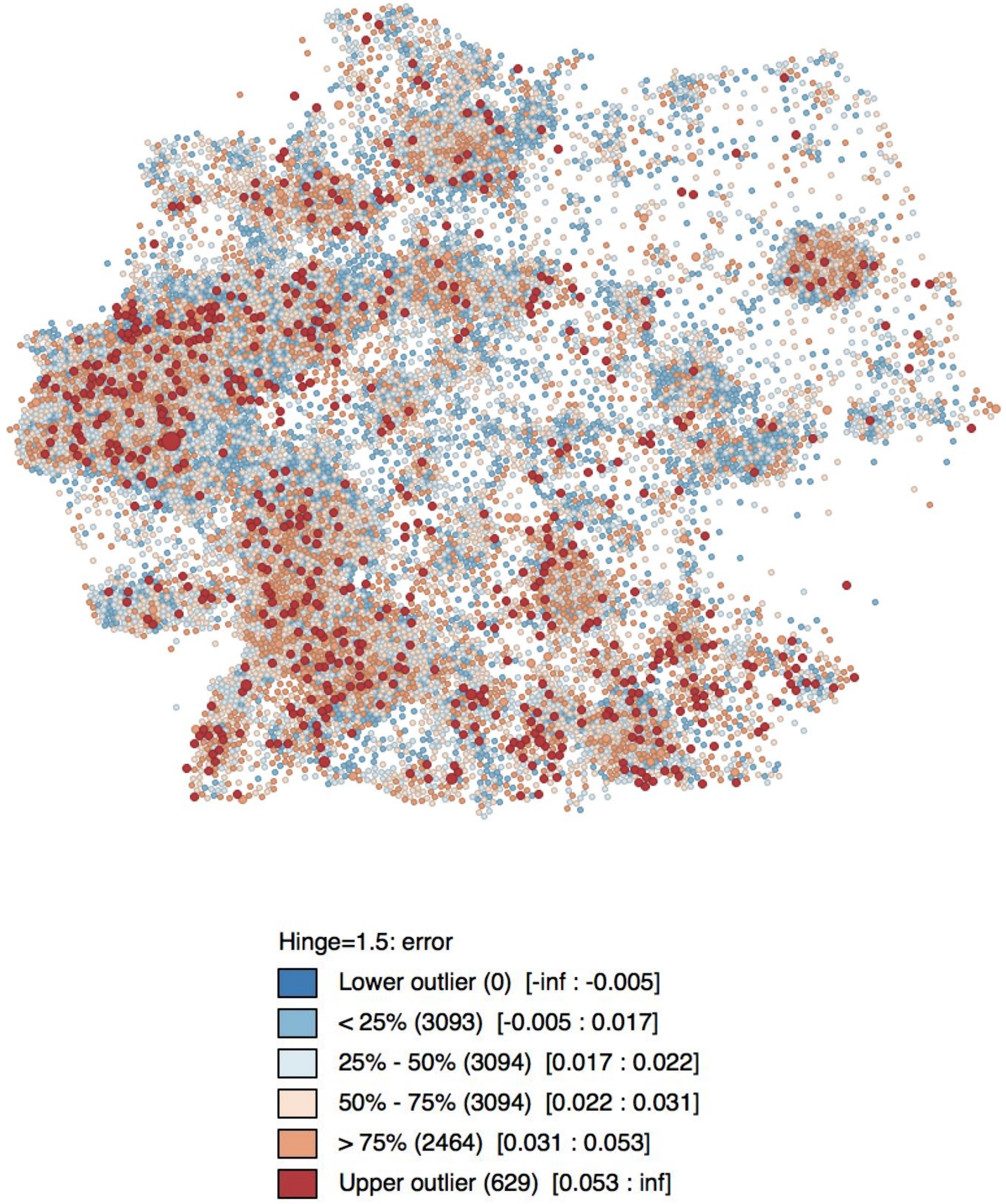
3a. Multiple Linear Regression



3b. Random forest model



3c. LSTM network


Hinge=1.5: error
🟦 Lower outlier (0) [-inf : -0.005]
🟦 < 25% (3093) [-0.005 : 0.017]
⬜ 25% - 50% (3094) [0.017 : 0.022]
⬜ 50% - 75% (3094) [0.022 : 0.031]
🟧 > 75% (2464) [0.031 : 0.053]
🟥 Upper outlier (629) [0.053 : inf]