Super Shoppers

Katya Donovan, Victoria McDermott, Noah Rivkin, and Samantha Young

Introduction

Super Shoppers will allow you to determine just when the price of any product, including oil and electricity, is cheapest.

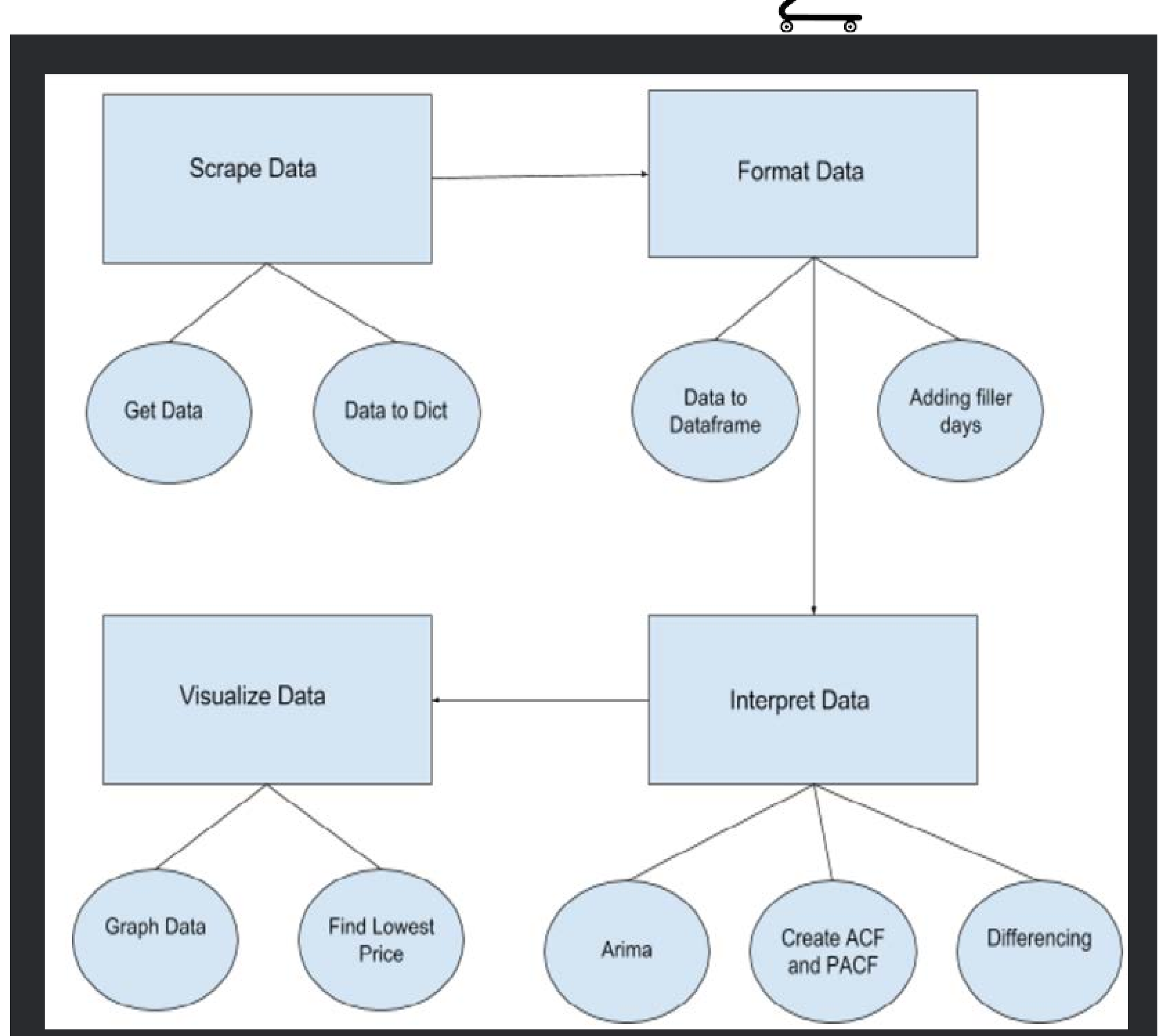
In addition to the obvious motivation of saving money, we wanted to explore this particular idea because we wanted to use statistical modeling and machine learning in order to make valuable predictions. We implemented a SARIMA model (Seasonal Auto Regression Integrated Moving Average) that takes current price values and fits them based on past price values. How SARIMA works is that it takes in a time series along with the parameters of the stationary time series and of the seasonality time series. In the end, the model is able to account for the time dependencies and the seasonalities in order to predict future prices.

With this model, our finished project is a website that allows users to investigate resources such as oil and electricity. With more data on product price history this website can be easily expanded to display the prices for any product. When the user interacts with the website they choose a product that they would like to learn more about. The website then indicates the cheapest days to buy this particular product.

Implementation

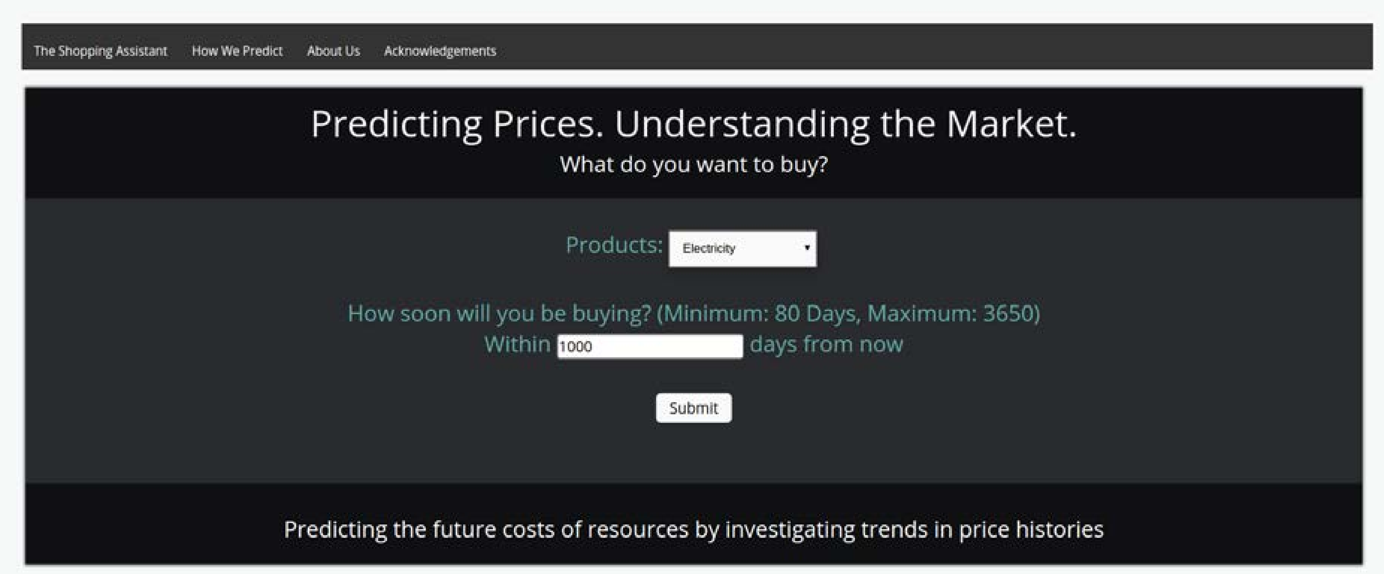
In order to make our code more manageable, we divided it into four classes: Collecter, Formatter, Interpreter, and Visualizer. As these names indicate, the Collecter class grabs the data from a source online and the Formatter class arranges the data into a dataframe that is then passed into the Interpreter class. The Interpreter then uses the data frame and passes it into the model, which predicts price values. These values are passed to the Visualizer which creates images of the predicted prices and past prices. When the user interacts with our web application, they choose a product and our application predicts the future price of that product at the specified time frame.

The following model demonstrates the layout of our web application.

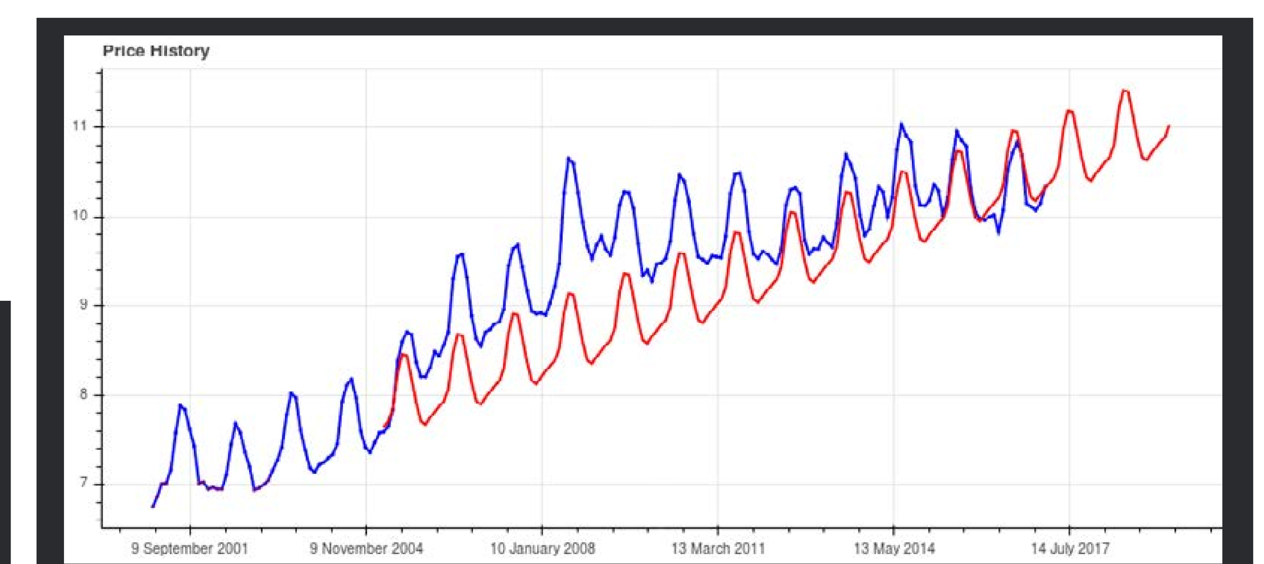


Results

For our project we created a website deployed on heroku which predicts future prices for various items using a seasonal ARIMA predictive model. Seasonal trends occur when there is a repeating pattern of an item being more expensive at a certain time of year. For example, winter coats would be more expensive in the winter than in the summer. Overall trends occur when the price of an item tends to go up or down as the years go by. We used seasonality and trends in items to determine when the best time for a person to buy a certain product is, both in terms of seasonal and overall trend variations. Our model outputs a visualization of the price graph. We included a hover tool which shows what the price is, the date that that price occurred, and whether that price is within 10% of the cheapest price. The following images display our interactive website.

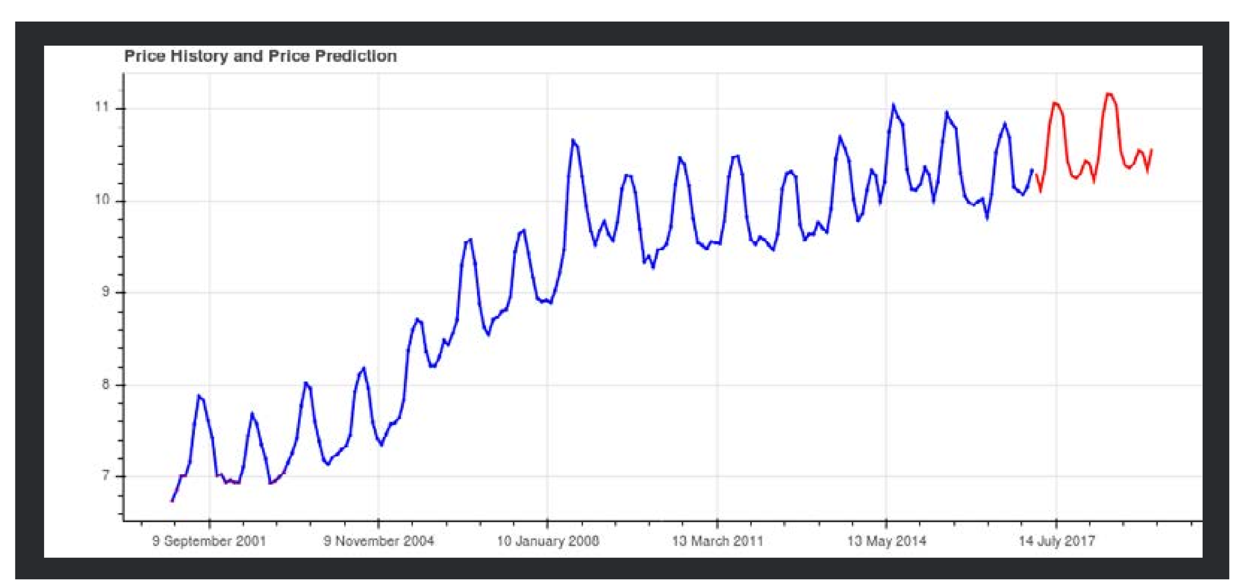


In order to test the accuracy of our model, we trained our model using only 100 days of data, and observed the fit of our model with the rest of the data set. This is displayed on the graph to the right. The red represents the predicted price, and the blue displays the true price of electricity.



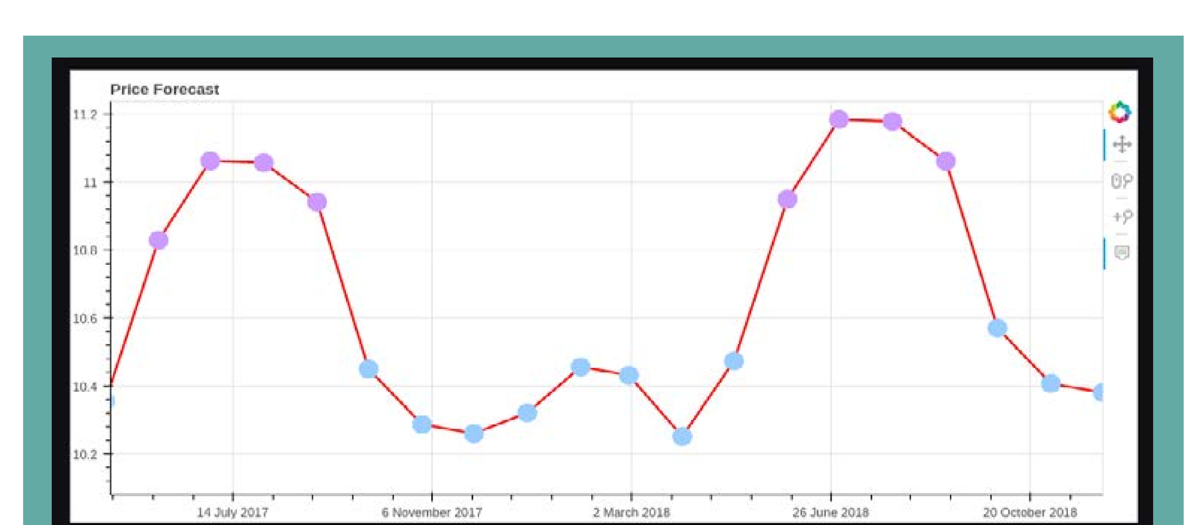
Predicting future electricity prices after a hundred days of data

Once we were confident that our model could predict accurately, we forecasted what the price would be for the future 26 months. The predicted and actual values are displayed in the graph to the left. Again, the predicted prices are in red, and the true price are in blue.



Predicting future electricity prices

The following graph shows what the user would see on the website, which contains only the predicted values for the amount of days that the user inputted. The purple nodes represent the prices are the lowest 5% prices for that range, whereas the blue nodes represent the highest 5% prices for that specific time period.

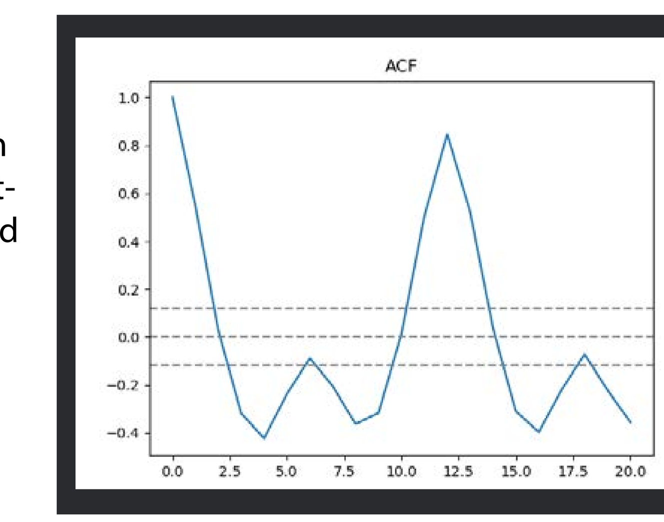


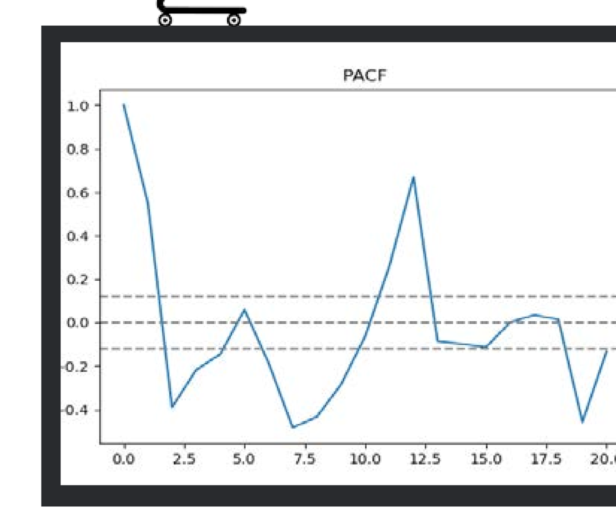
The website allows any user to determine future prices for certain products. This model can be used for any product, assuming there is significant amount of price history data.

The Seasonal Auto Regression Integrated Moving Average (SARIMA) model functions by finding the future prices based on the previous prices. Since product prices vary with season - for example, electricity prices change in respect to seasons - the model requires parameters for the seasonality and stationarity of our data. In order to account for this seasonality we decomposed the data using seasonal decomposition. Since SARIMA needs time series to be stationary, meaning that they were independent of time, we took the difference between the prices and their previous values. We also made the seasonality trend stationary by differencing it.

Once we had our stationary data and seasonality data, we found the parameters that would go into the model. We needed to determine the number of autoregression terms and lagged forecast errors for both the stationarity and seasonality data. One way to find the number of autoregression terms was by calculating, to the nearest integer, the first moment of the second standard deviation of the autocorrelation function for both the stationarity and seasonality data. In order to determine the lagged forecast errors we found the first time the second standard deviation of the partial autocorrelation function was reached. After calculating these values, we now had all the p and q parameters for our model.

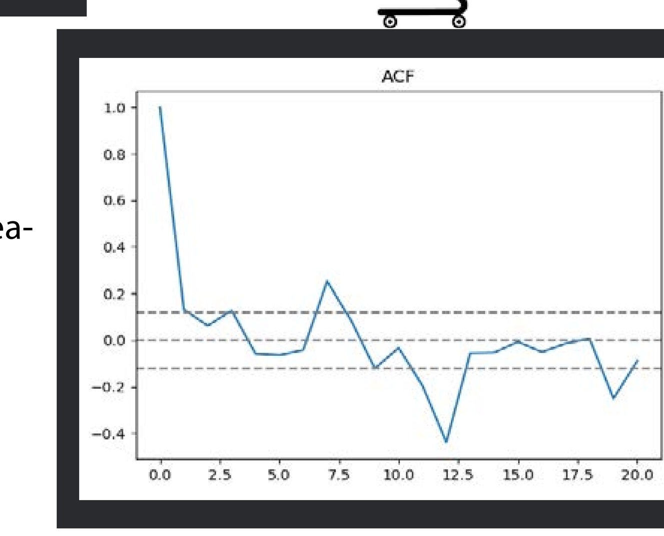
The autocorrelation function of differenced prices. The dotted lines represent the second standard deviation from the mean.

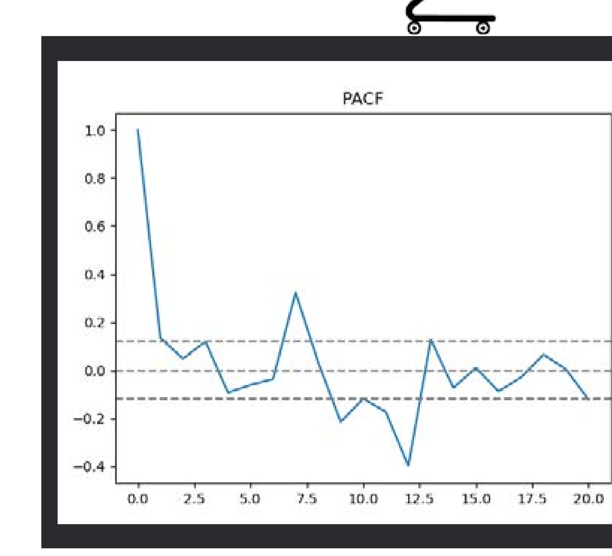




The partial autocorrelation function of differenced prices.

The autocorrelation function of seasonality in prices.





The partial autocorrelation function of seasonality in prices.

The Interpreter class then passes the data and its parameters into a SARIMA model which determines the prediction values for the requested number of days. This data is then passed into a Visualizer object which graphs the data, while also returning what day will have the lowest price. The script for our website then calls this function, in order to output the lowest price and the graph to the user.