Time Series Forecast of Gas Prices

Julia Troni, julia.troni@colorado.edu

Adam Manaster, Adam.Manaster@colorado.edu

INFO 4604/5604 Abel Iyasele

May 8th, 2023

Time Series Forecast of Gas Prices

Context

Whether we realize it or not, gasoline is a necessity for many people, particularly those who rely on their vehicles for transportation to work, school, or other daily activities. While gas prices are felt most at the pump, higher fuel prices have major impacts on the economy as a whole. If consumers spend more on gas, discretionary spending is often seen to decrease, which in turn has a "domino effect" throughout the broader economy. Beyond just consumers having less spending money (since gas is used in transportation), it affects the supply chain in every industry and can increase the price of virtually every available product.

Hence, accurately predicting gas prices can allow consumers and producers alike to make informed decisions.

However, the gas prices can be unpredictable and it is difficult to determine when they will change and by how much since the cost is influenced by a variety of global and local market factors, weather, natural disasters, and fluctuations in supply and demand.

Overall, the complex and interrelated factors that influence gas prices make it difficult to predict with precision. But, by monitoring trends in these areas and the effect on gas prices, we can make informed estimates about the direction of gas prices over time. Given our limited knowledge over this domain, we relied upon a publicly available dataset of historical retail gas prices, rather than considering the plethora of factors that impact gas prices (see "Limitations / Considerations" section for further discussion). We seek to apply best machine learning practices to model and predict future gas prices.

Time Series Forecast of Gas Prices

Problem Definition

Given a time series dataset of historical gas prices, the goal is to accurately predict future gas prices. By accurately understanding and predicting gas prices, individuals, corporations, and governments can better plan their budget and operations methods.

Methodology

Data Set

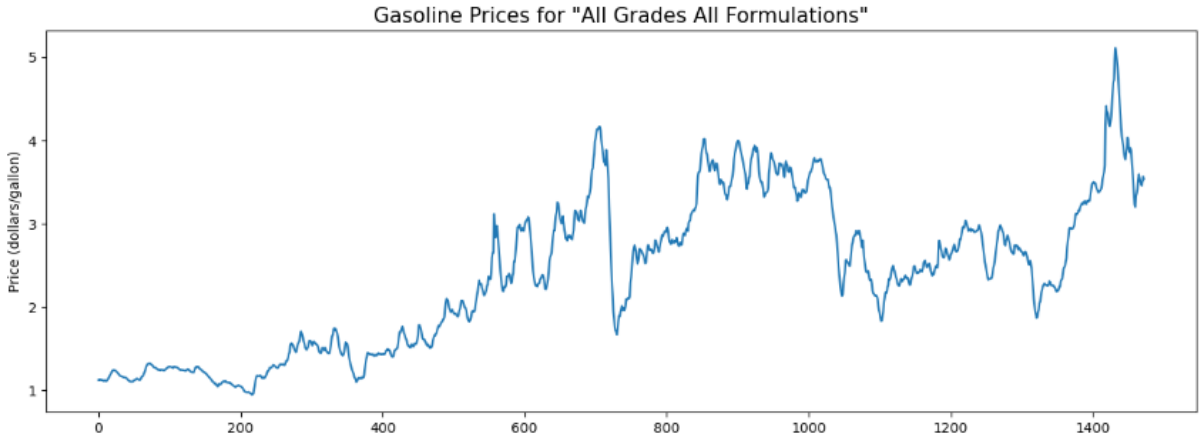
We used the dataset "Weekly Retail Gasoline and Diesel Prices in the US (Dollars per Gallon, Including Taxes)" found at https://www.eia.gov/dnav/pet/pet_pri_gnd_dcus_nus_w.htm. The data is sampled weekly from January 2nd, 1995 to March 20th, 2023 and contains 13 different categories, broken down by grade and formulation. Due to time constraints, we decided to predict prices for just one type of gas in this set: "All Grades All Formulations."

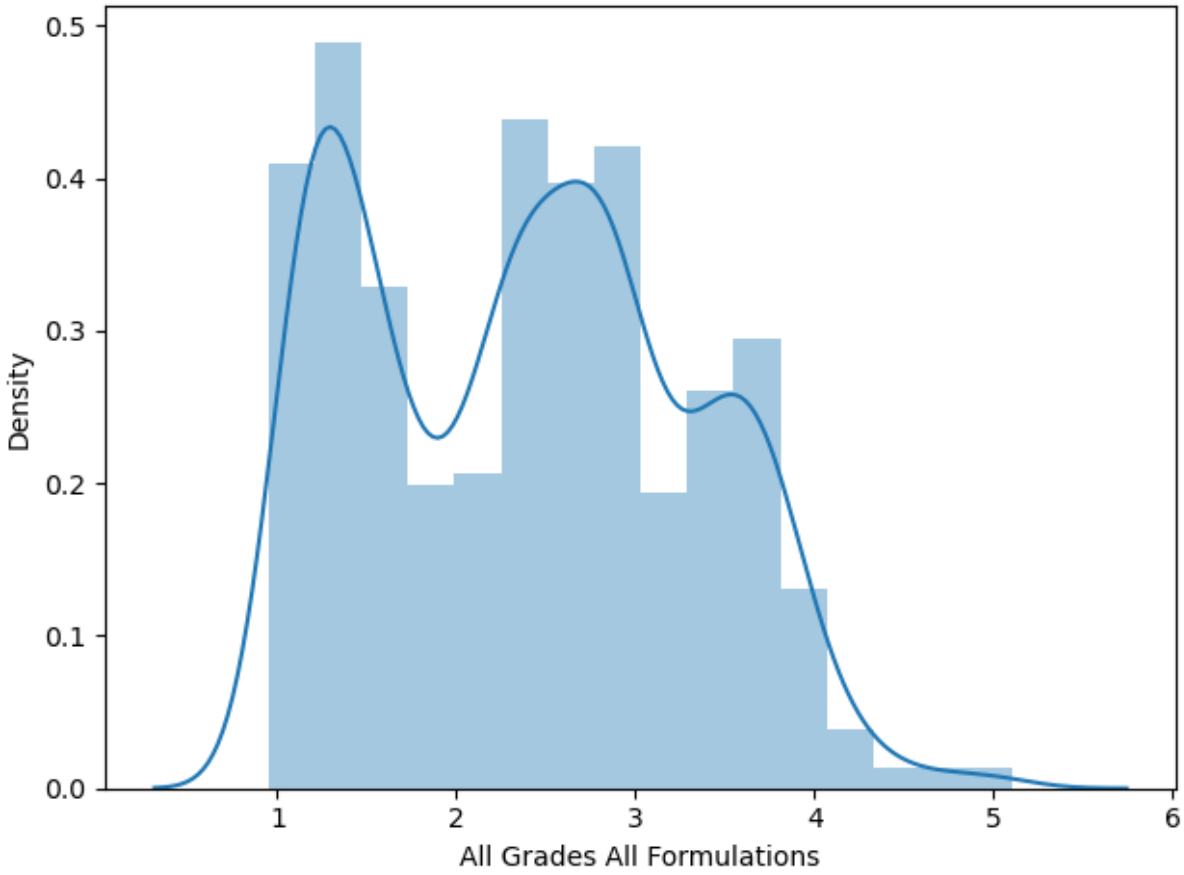
Data Analysis

First, as with all standard data analysis procedures, we began by cleaning and processing the data. Given our dataset, this involved removing unknown values and reformatting the dates which is necessary for time-series models. Below is a description of some basic statistics of the 'All Grades All Formulations' column of the dataframe. This is the data we trained and tested our models on:

	All Grades All Formulations
count	1473.000000
mean	2.386441
std	0.911106
min	0.949000
25%	1.511000
50%	2.403000
75%	3.025000
max	5.107000

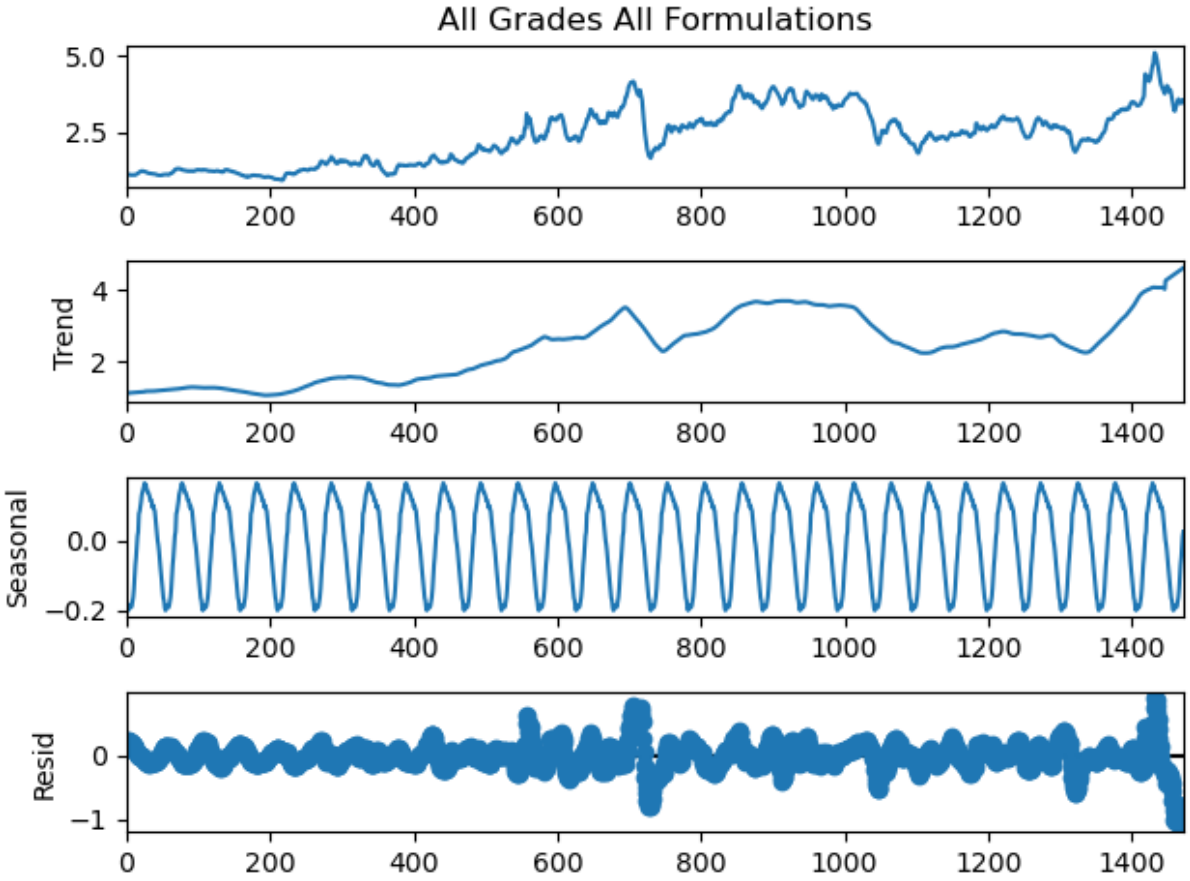
Visualizing the data was an important part of our analysis. Below is a glimpse of how this time series looks (note that the x-axis is formatted by observation number, not date), followed by a histogram detailing the relative density distribution of observations (created through "seaborn.distplot" in Python):

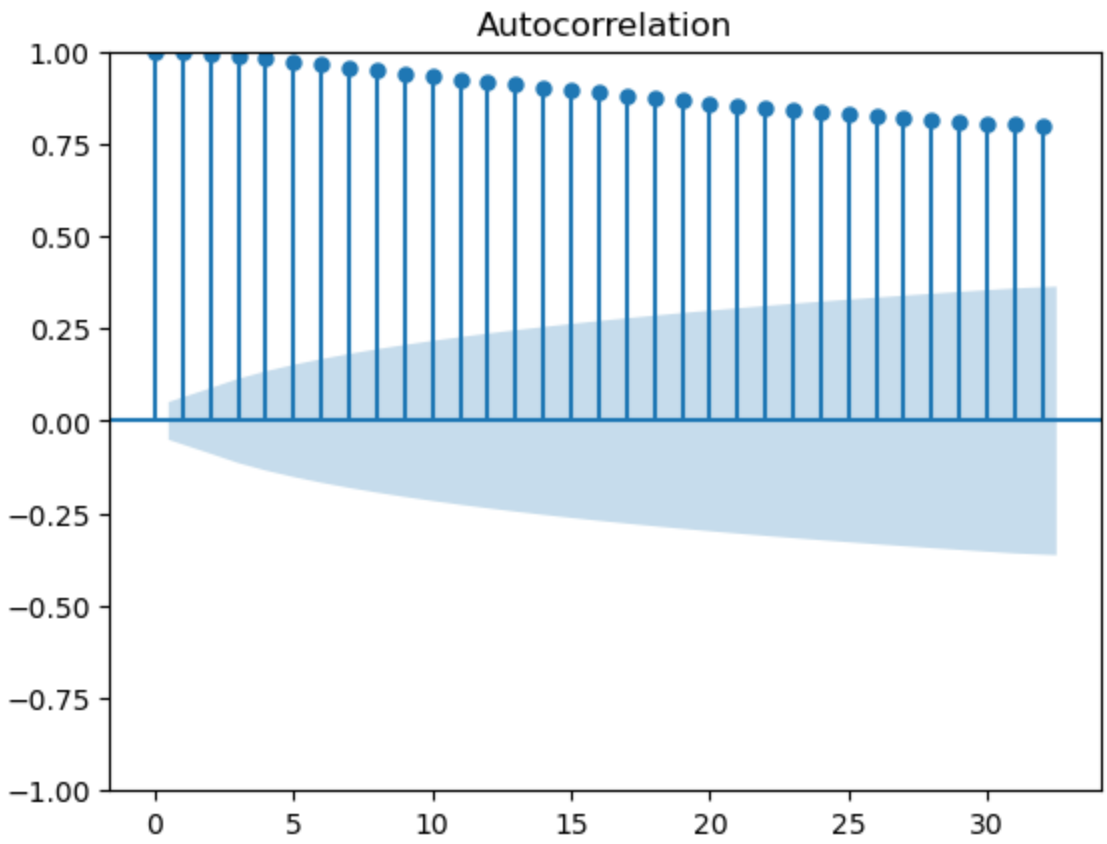




Lastly, since this is a time series dataset we needed to understand any trends or seasonality in the data. We utilized the “seasonal_decompose” function within the “statsmodels.tsa.seasonal” Python sub-package to observe the trend, seasonality (its repeating patterns/cycles), and residuals (remainder/error) of the data. Then, we used “plot_acf” from the “statsmodels.graphics.tsaplots” sub-package to plot lags on the horizontal axis and correlations on the vertical axis. These two plots are detailed below.

Time Series Forecast of Gas Prices





Model

We split our dataset such that 80% was used for training our model and 20% for testing - although it is recommended to save some data for validating the model, we did not have enough observations at our disposal (1,473 in total after data cleansing) and thus made the decision to focus our efforts on the rudimentary train/test sets. The data we chose to utilize were the weekly gas prices from January 2nd, 1995 - March 20th, 2023 for all grades and formulations of gas (as mentioned above).

First we implemented an ARIMA (Autoregressive Integrated Moving Average) model, which is commonly used to forecast future values for time series data. It captures the linear relationships between data points and the lags or differences between them. They are best used

when dealing with time series data to forecast future values based on historical patterns and trends. They can be used in a variety of fields such as finance, economics, sales forecasting, and environmental studies.

Specifically, ARIMA is best used when there is a need for short-term or medium-term forecasts and the time series data exhibits the following properties. There is evidence of autocorrelation in the data, meaning that the values are correlated with each other over time. There is a trend or seasonality in the data, which can be adjusted using the integrated component of the model. The data has already been preprocessed and cleaned, and any outliers or missing values have been addressed and it is stationary or can be made stationary through differencing. In Python, we implemented this model through the "statsmodels" package (specifically, "statsmodels.tsa.arima.model.ARIMA"). We performed a grid search to optimize its "order" hyperparameter - a tuple consisting of three components: lag order (the number of time lags), degree of differencing (how many times the data have had past values subtracted), and order of the moving-average model. The identified optimal values (in sequential order as described here) are presented in the "Description of Findings" section below.

Then, we decided to train and test a Long Short-Term Memory (LSTM) model to compare and contrast results from the aforementioned implementation. LSTM is a type of recurrent neural network (i.e., a deep learning technique) used commonly in processing sequential data, such as text, speech, and time series (which is our purpose). Centrally, it is designed to remember important information from the past, while simultaneously incorporating new information into the algorithm. The basic structure of an LSTM model consists of memory cells that incorporate several "gates" (those being input, forget, and output gates). These gates essentially control the flow of information into, within, and out of said memory cells. In Python,

we were able to implement this model through the “keras” package (specifically, we used the functions “Sequential”, “LSTM”, and “Dense” within “keras” to build the model). However, before we trained and tested the model, we had to scale the data down to a feature range between 0 and 1 (i.e., normalize the data). Then, to get an interpretable picture of the results, we inverted the predictions back to their original scale. Below is a brief snapshot of Python code detailing how we built and subsequently tested the model (note we utilized the ‘Adam’ optimizer, loss was dictated by Mean Squared Error (more on this later), and data were trained on 10 epochs with a batch size of 32):

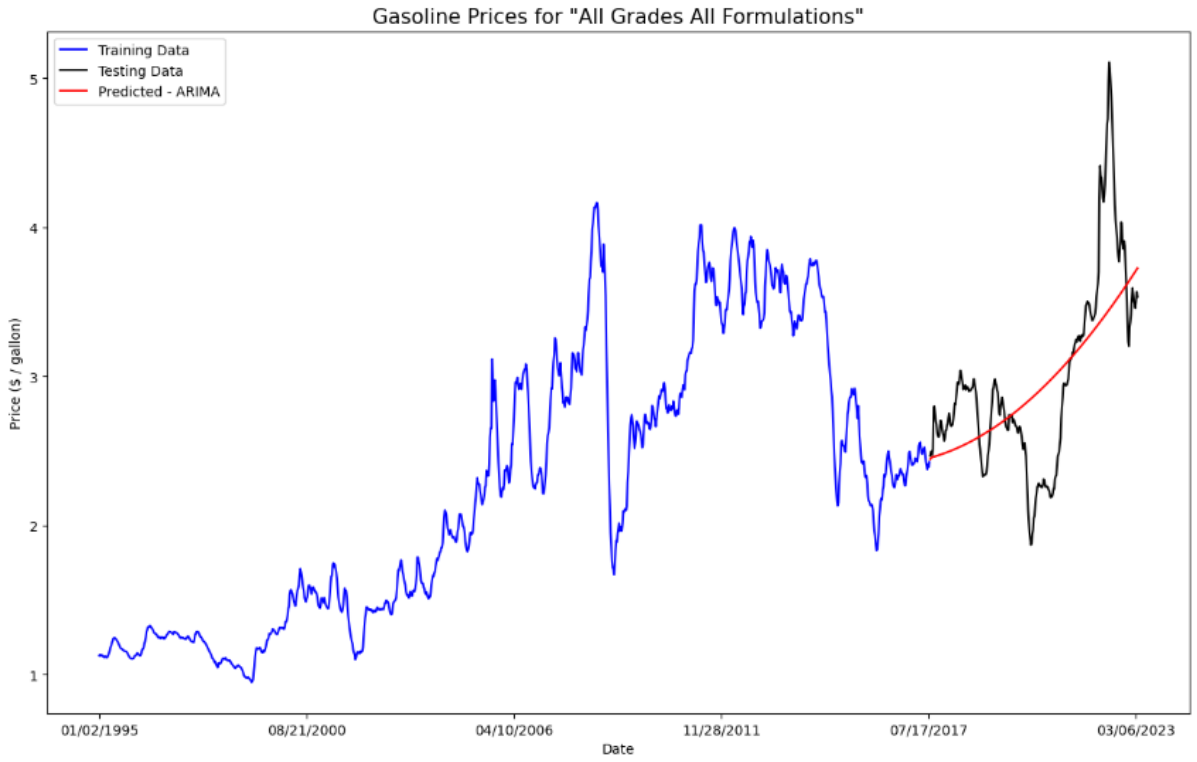
```
# build the LSTM model
model = Sequential()
model.add(LSTM(units = 50, return_sequences = True, input_shape = (X_train.shape[1], 1)))
model.add(LSTM(units = 50))
model.add(Dense(units = 1))
model.compile(optimizer = 'adam', loss = 'mean_squared_error')

# train the LSTM model
model.fit(X_train, y_train, epochs = 10, batch_size = 32)
```

Description of Findings

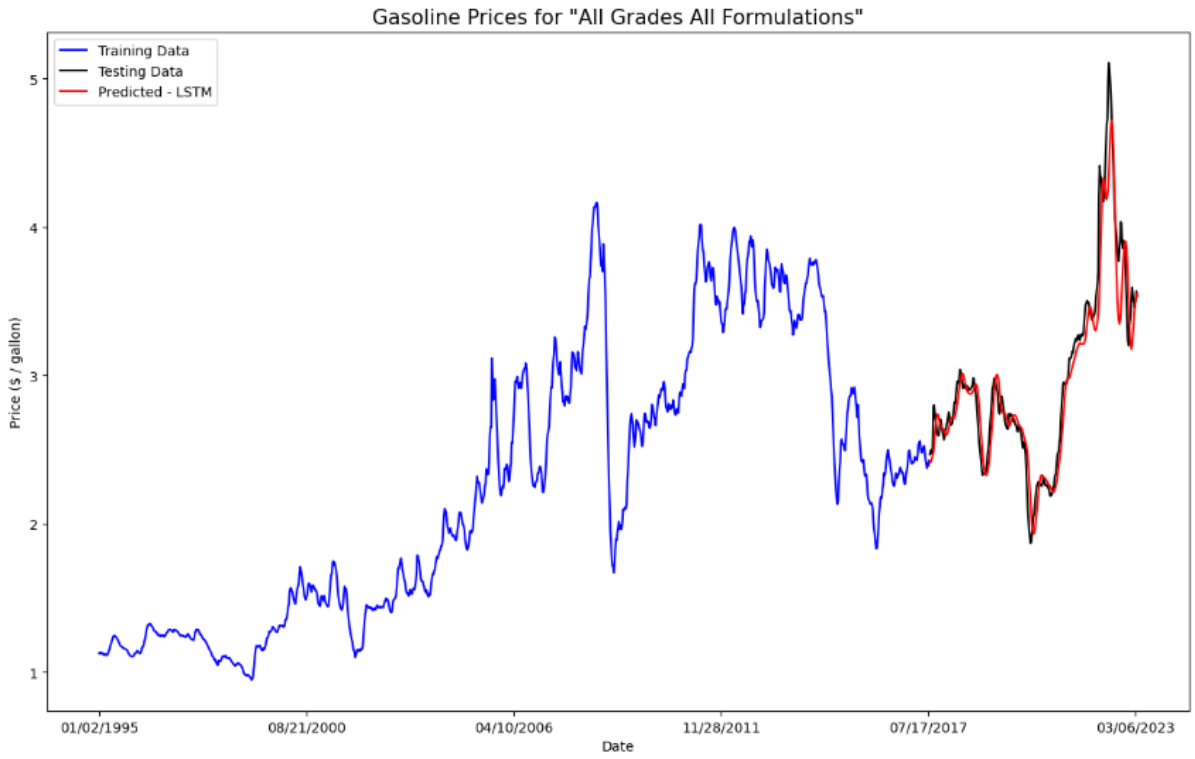
We used Mean Squared Error (MSE) as our evaluation metric for both models. The optimal order of the ARIMA model (2, 3, 7) (i.e., the order that minimized MSE) resulted in a value of 0.227. The MSE of the LSTM model we implemented had a value of 0.034.

Clearly, as shown in the figures below, like any machine learning problem, there are issues with both models. The ARIMA model contains an integrated “smoothing” effect that does not capture realistically the noise and fluctuations of the data. Although it does a fine job capturing the general trend of our data (I use the term “general” loosely here), it would not be

adequate for predicting gas prices well into the future (the curve shown in red would keep continuing on its upward trajectory).



For the LSTM model, it is quite obvious that it is overfitting the data (again, red curve) - the most common problem in machine learning. Thus, this model would not generalize well to new data (i.e., it is not generalizable). We recognize and understand the pitfalls of this implementation, but given our more elementary knowledge of this algorithm, we found it difficult to pinpoint how to improve upon the LSTM model.



Real-World Applications

While this is just one application of such prediction modeling, similar models can be used for forecasting other trends. ARIMA and LSTM models are powerful tools for forecasting trends in a variety of applications beyond their original use in time series forecasting. These models can be applied to financial and economic forecasting, energy demand forecasting, traffic forecasting, environmental forecasting, marketing forecasting, and healthcare forecasting. Financial and economic forecasting can benefit from ARIMA and LSTM models in predicting stock prices, exchange rates, and future sales for a product or service, which can help businesses with inventory management and resource planning. Energy companies can use ARIMA models to forecast electricity demand to manage their supply and demand balance. Traffic forecasting can help transportation planners optimize traffic flow and alleviate congestion. Environmental

forecasting can aid in environmental management and disaster response by predicting weather patterns, air pollution levels, and water quality. ARIMA and LSTM models can also help businesses optimize their marketing strategies by forecasting website traffic, social media engagement, and other metrics. Finally, healthcare forecasting can aid in resource planning and healthcare policy decision-making by forecasting hospital admissions, patient outcomes, and disease incidence.

These are just a few examples of the many applications of time series forecasting models in various fields. As the amount of available data continues to grow, the use of time series models is likely to become even more widespread.

Constraints and Future Work

One of the most notable limitations in our model is the dataset we used - it included solely weekly gas prices of several different grades and formulations and had just shy of 1,500 data points (i.e., not much data for training / testing / validation). We chose to implement machine learning models that are most suitable for forecasting time series, purely because we do not have the domain expertise to know what other variables could be suitable predictors for gas price(s) (considering that it is a complex problem with many confounding layers as discussed in the Context section). Thus, we chose to utilize a dataset that only required time series analyses. However, for more comprehensive analyses in the future, we could consider a more “vast” range of data to determine how economic and geo/socio political issues feed into the rise and fall of gas prices globally.

Another (more personal) challenge we faced in this project was that one of our group members opted to take Exam 1.5, bailing on our group project and he did not notify us until after

the fact. Although we understand and respect this person's decision, it left us with more work than anticipated and thus it became a more time-consuming endeavor.

As for all projects, time constraints and pressures played a large role in many of our decisions throughout the project. We did not have sufficient time to improve and adjust our models as we would have wished to. In future iterations, we would like to implement and compare different models, tune parameters, and use cross validation.

Conclusion

Overall, our results for both models (ARIMA and LSTM) show promising results, considering the low MSE values we obtained upon testing (0.227 and 0.034, respectively). ARIMA is most aptly-suited for short-term forecasting and LSTM is a powerful network that is great for time series predictions, taking into account its robust approach to "past" data. However, we also realize the downfalls of both implementations (namely, ARIMA is too smooth a forecast and is not great ideal for capturing the "noise" of the data, and LSTM overfit the data and thus would not be considered generalizable given our current implementation). As discussed above future work remains to be done - however, this is a promising start.

References

1. https://www.eia.gov/dnav/pet/pet_pri_gnd_dcus_nus_w.htm
2. ARIMA Model - Complete Guide to Time Series Forecasting in Python | ML+ (machinelearningplus.com)
3. Introduction to Time series Modeling With -ARIMA - Analytics Vidhya
4. Stationarity | Statistical Tests to Check Stationarity in Time Series (analyticsvidhya.com)
5. Netflix Stock Prediction with ARIMA | Kaggle
6. Time Series Analysis- Part II. In the previous blog we learnt about… | by Madhu Ramiah | Medium
7. Time Series in Python — Part 2: Dealing with seasonal data | by Benjamin Etienne | Towards Data Science (medium.com)
8. https://www.capitalone.com/tech/machine-learning/understanding-arima-models/
9. https://intellipaat.com/blog/what-is-lstm/?US
10. https://www.mathworks.com/discovery/lstm.html