

# MSBA Capstone

## Maverik Team 1: Data Analysis Report

Time-Series Forecasting Candy Bar Sales  
April 25, 2021

Sam Erickson | Tim Gain | Kelsey Metivier | Willem van der Schans

GitHub:

<https://github.com/Kydoimos97/CapstoneMSBA2021>

Dropbox:

[https://www.dropbox.com/sh/i0n97zm0yexbg4f/AABNZB4SAILaxHla\\_i7kdLpDa?dl=0](https://www.dropbox.com/sh/i0n97zm0yexbg4f/AABNZB4SAILaxHla_i7kdLpDa?dl=0)



## Table of Contents

Executive Summary.....	6
Overview	6
Challenges	6
Solutions	6
Highlights	6
Introduction.....	8
Project overview	8
Business Problem	8
Primary Objective	8
Summary	9
Outline	9
Data Preparation.....	10
Overview	10
Outliers	10
Missing Data	11
Restrictions.....	12
SQL Packages	12
Computing Limitations	12
Implementation.....	13
ARIMA	13
Inputs	13
Seasonality	13
Application .....	15
Overview	15
Fail-Safes	15
Results.....	17
Inputs	17
Global performance	17
Extremity performance	17
Conclusion .....	19
Appendix.....	20

Appendix A	20
Appendix B	20
Appendix C	21
Appendix D	21



# Executive Summary

## Overview

Maverik would like to improve the sales forecasting model of candy bars at their convenience stores. The purpose of this Capstone data analysis is to create a time-series forecasting model for Maverik to accurately predict candy bar sales at each of its convenience store locations. As a result, company management can better optimize resources and increase profitability by predicting item-level candy bar sales at each of its convenience stores, projecting ten days into the future.

## Challenges

Maverik provided almost three years of candy bar sales data containing over 1.3 million rows of data. The amount of data presented a few challenges:

1. Requiring granularity on a large matrix size caused computational limitations
2. Maverik SQL databases are only compatible with specific packages in R and Python
3. Data contained large outliers
4. Pricing data likely contains errors (e.g., \$0.10 Tootsie Rolls selling for \$51.00)
5. 11,799 NA's in the minimum and maximum temperature fields
6. Seasonality and lag

## Solutions

To overcome the issues above, we utilized the following solutions:

1. Sub-sampling loops provide granularity on data with more manageable sizes
2. Utilized Python for SQL compatibility and efficiency
3. Four standard deviations identified 12 extreme outliers, which were removed
4. Corrected pricing data
5. Interpolation to impute missing data points

## Highlights

Our team created an application that allows Maverik management to generate a predictive model based on given inputs. The application's output produces the best out-of-sample RMSE error metric to show how accurate the model is and saves the result in CSV files for future reference. The application allows company management to customize forecasting depending on specific needs or queries. By accurately predicting specific candy bar sales at specific store locations, the company can optimize resources by ordering the optimal number of candy bars for each of its sites, thus improving company profitability.



# Introduction

## Project overview

Maverik is one of the most well-known names in gas stations and accompanying convenience stores throughout the intermountain region. Their website indicates they are located in "more than 350 locations across 11 western states, making it the largest independent fuel marketer in the Intermountain West"<sup>1</sup>. Although Maverik has been very successful with both its fuel and convenience store sales, the company would like to improve the sales forecasting model of candy bars sold in its convenience stores. This project seeks to provide a reliable time-series forecast that the company can use to predict the sales of specific candy bars within each of its stores.

## Business Problem

Maverik requires reliable forecasting models for the inventory carried in its convenience store locations to place restocking orders and generate financial forecasts. Numerous factors affect individual stores' or groups of stores' calculations of these forecasting models, including location, size of the store, day of the week, holidays, weather, and many more variables. Maverik also has the challenge of determining whether they fulfill customers' needs by carrying the items that Maverik customers want. Accurate forecasting models allow the company to better allocate resources by ordering and maintaining the optimal quantity of each candy bar to increase company profitability.

## Primary Objective

The primary objective of this analysis was to build a time-series forecasting model to predict candy bar sales at the item\_id and site\_id levels. Maverik provided over 1.3 million rows of historical sales data spanning from November 2017 to September 2020. The end goal of this project is to enable better management of candy bar inventory, pricing, budgeting, and forecasting.

---

<sup>1</sup> <https://www.maverik.com/about-maverik/>



## Summary

While the initial business problem and objective seemed straightforward, the team encountered several obstacles illuminating the project's complexity. The size of the data set proved to be the main roadblock during the project's lifetime. Without access to the computing power necessary to perform a time-series model on the entire data set and the requirement to maintain granularity, we opted to use dynamically-generated data subsamples to feed to an ARIMA model. ARIMA is a time-series specific model where the algorithm's assumptions consider the unique attributes of time-series data, often producing more robust results than a simpler linear regression model would with less assumption violation. To make our model user-friendly, we built a stand-alone program with a GUI in an executable file that allows the end-user to perform predictions without having to interface with the code or install Python. The program focuses on transparency and saves all output in CSV for record-keeping and further analysis.

## Outline

The report will first explore our data preparation, providing an overview and deep-dive into outliers and missing data. Next, the restrictions and limitations of the project will be discussed, followed by the implementation of our time-series model. Finally, the report will detail the application and our final product.

# Data Preparation

## Overview

The data set contained historical data points from a subset of Maverik's convenience stores from 11/08/2017 to 09/05/2020. There were 17 variables and 1.3 million rows of data. The data is separated by day per candy bar sold.

## Outliers

The distribution of Sales and Quantity Sold in the Maverik data was heavily skewed to the right. The majority of customers (77.5% of transactions) purchase only 1 or 2 candy bars with total sales of \$4.99 or less. However, there were six instances when the quantity sold exceeded 1,000 candy bars in one day. One additional instance shows the quantity sold exceeding 10,000 at one time. These data points may not be accurate because they are extreme compared to the typical transaction.

The methodology used in determining and removing outliers in this data set was based on comparing the actual sales of each candy to the mean sales of that same candy bar. Any sales entries that were four or more standard deviations from the mean sales were considered an outlier and, therefore, removed from the data. In utilizing this approach, only the 12 most extreme items (0.0009% of the data) were removed from the analysis. We chose to use four standard deviations to account for possible discounts or price surges that allow more considerable differences between the mean price and actual daily sales of a specific Item and Site combination.

Additionally, an item-specific outlier was spotted; namely, Item\_ID -12066 (TTS TOOTSIE ROLL \$.10) appears as though it should cost \$0.10 each based on the name of the product. However, there are several instances where the price exceeds \$5.00 and even goes as high as \$51.00, thus skewing the sales data for that item. To solve this issue, the price value of \$0.10 was imputed for item\_id -12066 to match the named description, making the data more accurate and reliable.

A comparison between the originally supplied data and the cleaned data is shown in figure 1. Note that the before visualization already included the price fix for item\_id -12066

## Missing Data

The 'maxtemp' and 'mintemp' variables were each missing 11,799 values. We could not pinpoint a single factor for why this temperature data was missing as the NA's were randomly scattered across dates and site ids. Early on in our analysis, we considered removing the temperature variables altogether; however, after consultation with Maverik, we learned the company plans to utilize forecasted weather data. The interpolate function from the Scipy package was used to impute the NA's. Interpolate uses a linear function and the nearest data points to impute the missing data points.

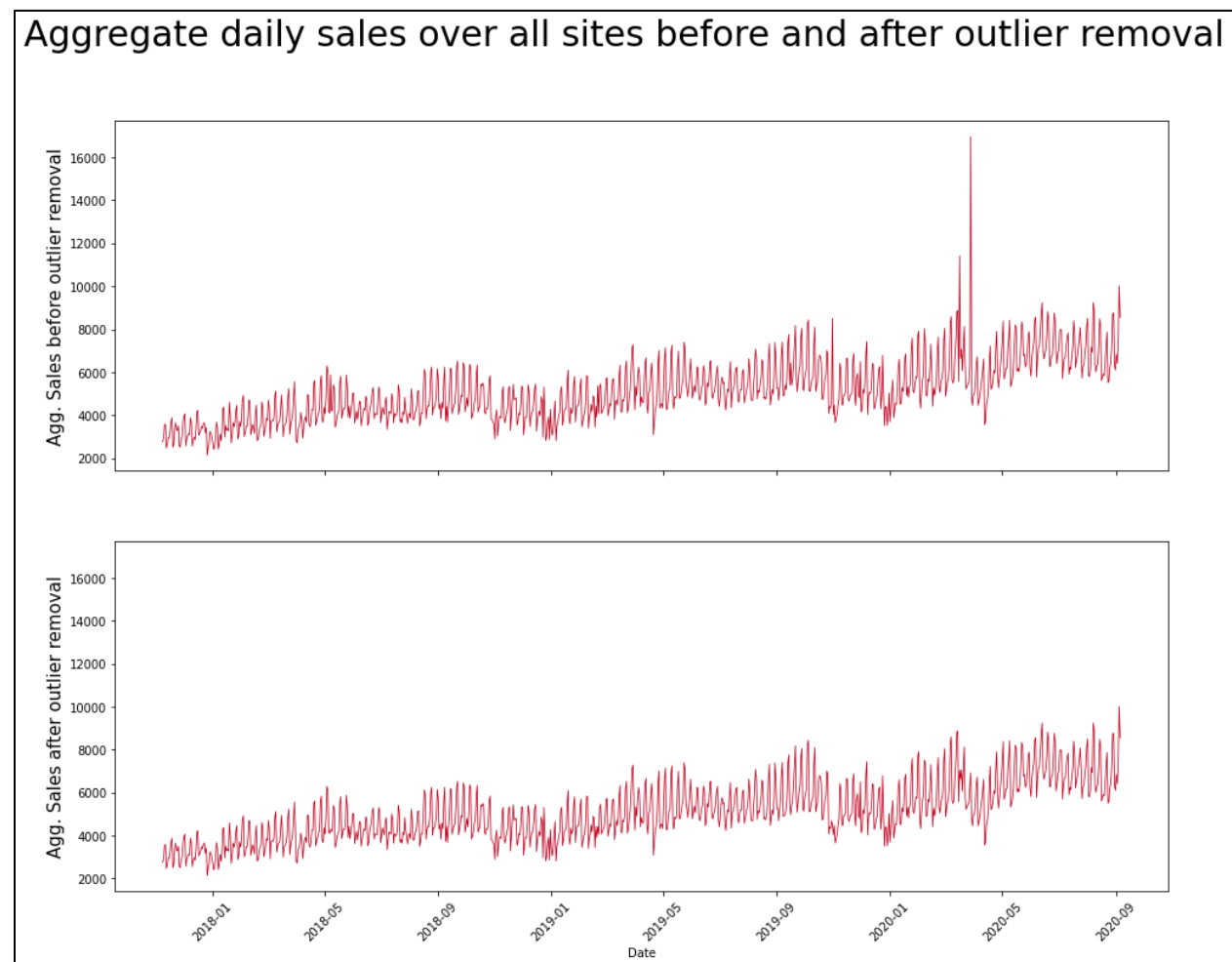


Figure 1: Original Data vs. Cleaned Data Comparison

# Restrictions

## SQL Packages

Maverik limited the use of packages in R or Python to those compatible with their SQL database. The team decided to use Python as there were more compatible packages available that allowed for time-series analysis. Using Python provided an additional challenge as collectively, the team was more experienced with R. Python was used to perform all data operations - cleaning, initial analysis, and time series analysis. Two groups of packages were used the essential packages that ensure the functionality of the code and the optional packages that merely provided quality of life improvements to the developed application. Application-specific uses can be found in the main code deliverable. The used packages are outlined below:

### SQL Compatible (Essential)

- Pandas
- Numpy
- Statsmodels
- Scipy
- Sk-Learn
- Matplotlib
- Math
- Tkinter
- sys

### Unknown Compatibility (Optional)

- Subprocess
- Threading

## Computing Limitations

Computational power problems occurred throughout the project due to the size of the data matrix. As required by the project's scope, maintaining granularity in the data was crucial for producing predictions at the item level and eliminated the option of dimension reduction or aggregation of the data into a more manageable size. Subsampling loops were created to perform time-series predictions in a manner our computers were able to handle. Performing a time-series model on the entire data set without access to the necessary computing power would have taken several days, rendering this option infeasible.

# Implementation

## ARIMA

Our time-series analysis uses the ARIMA (autoregressive integrated moving average) model. ARIMA was explicitly built for time-series modeling, and the assumptions in the algorithm apply to the unique attributes of time-series data. The algorithm is versatile and capable of handling both univariate and multivariate data. ARIMA was critical for this project, as the computational power required is input-based, allowing for flexibility to adjust the input variables based on the computing power available and enabled the team to run models with limited RAM. However, it should be noted, that while computational power required is based on inputs, so is the performance of the model.

## Inputs

ARIMA requires three inputs:  $p$ ,  $d$ , and  $q$ .

$p$  defines the number of autoregressive terms, which are the number of immediately preceding values in the series. These initial values are used to predict the value at present.

One stipulation of using ARIMA is that the data needs to be stationary, meaning the data should not show a trend of growth or decline,  $d$  is the number of nonseasonal differences required for the data to be stationary. With  $d$  being able to be grid searched, no adjustments have to be made to ensure stationarity within provided data to the ARIMA model allowing for simplification of feature engineering.

Finally,  $q$  defines what is taken into account for the Moving Average and is the number of lagged forecast errors in the prediction equation.

Our final application allows for grid searching of the ARIMA model, which selects the best  $p$ ,  $d$ , and  $q$  for any given data set by iteratively analyzing RMSE based on a  $p, d$ , and  $q$  configuration grid provide the best results.

## Seasonality

ARIMA assumes no or very little seasonality to be present in the provided data. Figure two shows a subsample of 4 item and site combinations. As shown, little seasonality occurs. Very few items, however, like the tootsie rolls, do show seasonality. SARIMAX and extension of ARIMA can deal with seasonality, but we did not find time to implement this approach yet.

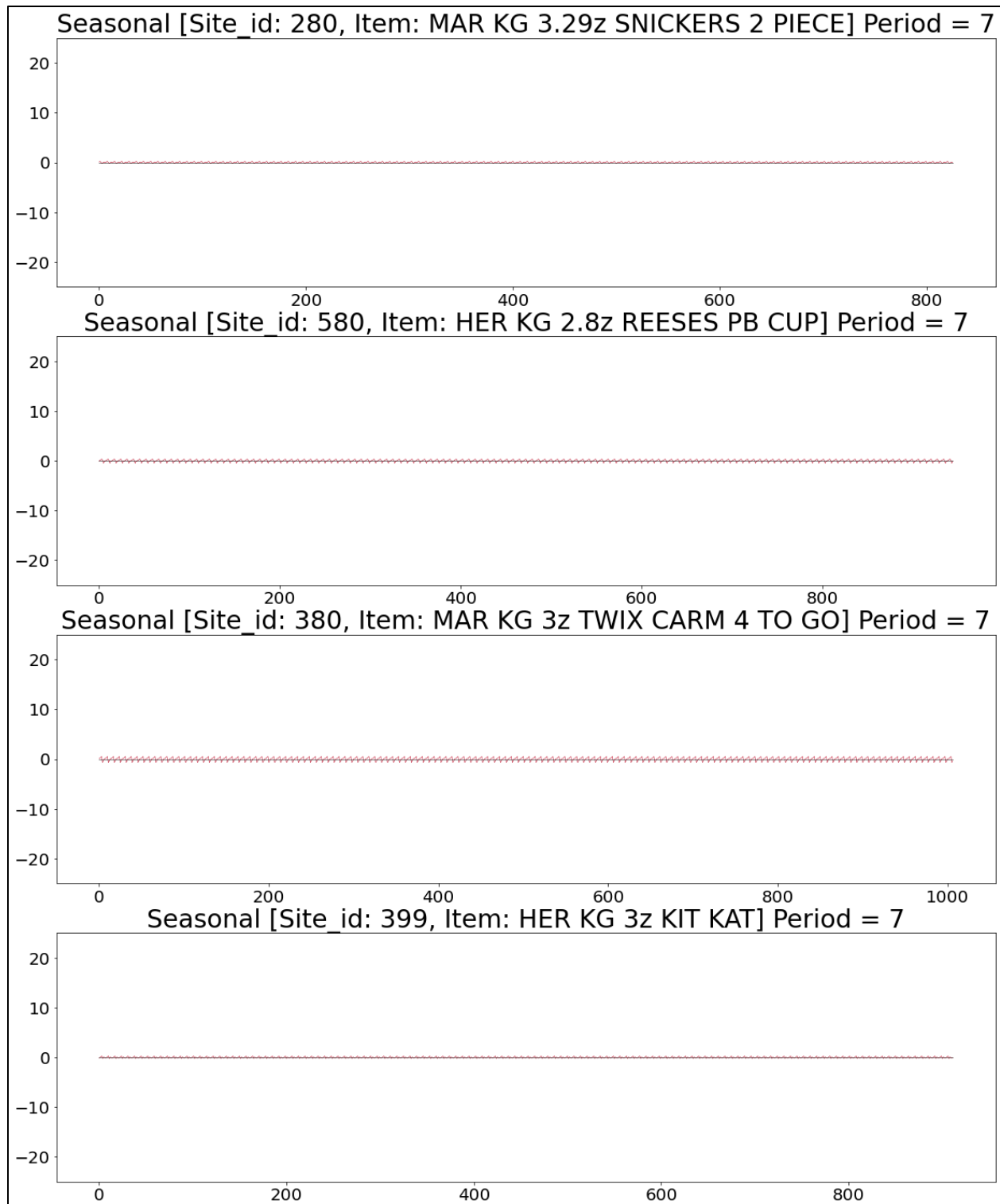


Figure 2: Seasonality of Sub Samples

# Application

## Overview

To solve Maverik's need to predict item-level candy bar sales at each of its convenience stores, our team's analysis resulted in creating an application which company management can use when placing new orders for candy bars and maintaining existing inventory. The application is customizable, allowing management to predict individual candy bar sales at each of its locations based on historical sales and other relevant time-series variables. The user may choose various factors, including sales forecasts for certain specific candy bar item numbers or forecasting the top N number of most-sold items. Depending on the need, the user may predict sales for all locations or drill down to a specific store number. The individual p, d, and q parameters of ARIMA models may be selected based on need. The user may also decide on how many days of data should be reviewed when building future predictions. As with many predictive models, using inputs that consider more data will often lead to a more accurate forecast, but the model will require a lot more time and machine processing power to run. Once all the input parameters have been entered, the option is given to save the data in a CSV file which may be referenced later if needed. The application's output will also display the forecast's Root Mean Square Error (RMSE) on the test set within the data to show how accurate the model predicts unknown future values. The GUI can be seen in figure three. Further information can be found in the documentation, which will be provided at a later date.

## Fail-Safes

The application also has a few "fail-safes" built into it. Error message boxes are used to help direct the user to the correct usage of the application. For example, if the user inputs a site id that is not present in the dataset, the application will throw an error stating that "Model failed for all Item\_ID and Site\_ID combinations." Fail safes are essential to make sure the application does not need to be babysat while running. Stress testing the application has uncovered many potential breaking points, which are all accounted for. All errors are coerced to a CSV file regardless of the output option being selected as yes. CSV examples can be seen in Appendix B and C. Error message examples can be seen in Appendix D.

Maverik: Candy Bar prediction

MAVERIK

Files and Directory

Main DataframeBlack/Output/CapstoneMainDF.csv
Product DataframeBlack/Output/ProductsDF.csv
Output DirectoryDesktop/Black/Output

General Inputs

Item\_ID2CheckTop\_N ✓

Site\_ID280,380CheckSite\_ID List ✓

Outlier Std.Dev.4Check✓

Days to Predict10Check✓

Parameter Inputs

Starting p7p Grid Mult0,1,2

Starting d0d Grid Sum0,1,2

Starting q1q Grid Mult0,1,2

Options

Gridsearch AllNoGridsearch d onlyYes

Save OutcomeNoShow WarningsNo

Submit and Run

Exit Program

Total time elapsed = 18 Seconds

Models evaluated = 12

Output

{Item\_ID = -14148 | Site = 280  
Item\_Name = MAR KG 3.29z SNICKERS 2 PIECE  
---Prediction---  
2020-09-06 1.67  
2020-09-07 1.92  
2020-09-08 1.70  
2020-09-09 1.76  
2020-09-10 1.82  
2020-09-11 1.82  
2020-09-12 1.83  
2020-09-13 1.85  
2020-09-14 1.86  
2020-09-15 1.87  
Freq: D  
-----  
Predicted Sum of Quantity 10 Days = 19  
Expected Sum of Quantity last 10 Days = 18  
OOS Total Error = -7  
Best OOS Config = (7, 0, 1)  
Best OOS RMSE = 1.36  
  
} {Item\_ID = -14148 | Site = 380  
Item\_Name = MAR KG 3.29z SNICKERS 2 PIECE  
---Prediction---  
2020-09-06 6.60  
2020-09-07 5.90  
2020-09-08 6.37  
2020-09-09 6.33  
2020-09-10 6.80  
2020-09-11 7.33  
2020-09-12 6.87  
2020-09-13 6.62  
2020-09-14 6.44  
2020-09-15 6.45  
Freq: D  
-----  
Predicted Sum of Quantity 10 Days = 66  
Expected Sum of Quantity last 10 Days = 72  
OOS Total Error = -29  
Best OOS Config = (7, 0, 1)

Figure 3: Application GUI after A completed prediction run



# Results

## Inputs

A stress test was completed with a list of fifty items overall sites included in the data set. The items were picked based on their sales data. Two critical points of sales were identified in which the program would behave differently; minimal sales and maximal sales. This prompted us to select the ten least and most sold items and add ten items surrounding the 1<sup>st</sup> quartile of sales, mean sales, and the 3<sup>rd</sup> quartile of sales. This list of fifty items can be found in appendix A.

The parameter inputs are set to default and were chosen to limit total runtime and ensure a prediction that was reflective of potential model performance. The input options can be found in appendix figure 4. As shown, grid searching is allowed, and the outcome is saved. Warnings are turned off, so the program doesn't require any user input while running.

## Global performance

The program performed well without running into any unforeseen errors. 13500 Models were considered, and of those models, 10420 ran. One hundred sixteen combinations ran into errors causing not all considered models to be completed. Of these 116 errors, 94% were invalid combinations of site\_id and item\_id, 2% were lu decomposition errors, which is a bug with the stats models package, and 4% were historical data errors where not enough historical data was present to make a prediction. The total runtime of the program was 2 hours and 36 minutes with an average ram usage of 805Mb and 30% CPU utilization (6 core 3.70 GHz). The mean hold-out RMSE is 1.02 overall items, and with 0 RMSE predictions removed 2.13. The program predicted a total quantity of 6081 items to be sold in the next ten days over 5223 sold in the last ten days. Note again that RMSE can be significantly improved by increasing the range of p and q grid searching; we didn't do this due to time restrictions on runtime.

## Extremity performance

Both the least and most sold items are extremities that we want to further outline in this paper. First, the most sold items. This list of 10 considered 2700 models, and the program completed all of them, meaning no errors occurred. The mean hold-out RMSE is 3.22 per item. The total predicted quantity is 5931 items over 5263 items in the last ten days. The total runtime for this list was 41 minutes. The least sold item list also considered 2700 models, but only 216 were completed. The program predicted 0 sales for all items included and ran into 92 errors. All errors were handled correctly and are presented in an error CSV. The runtime for this section was only 3 minutes, showing the impact of efficiently coercing errors and moving on to the next item.

General Inputs			
Item_ID	-12172,-29955,-21234,-	Check	Item_ID List ✓
Site_ID		Check	All_Sites ✓
Outlier Std.Dev.	4	Check	
Days to Predict	10	Check	

Parameter Inputs			
Starting p	7	p Grid Mult	0,1,2
Starting d	0	d Grid Sum	0,1,2
Starting q	1	q Grid Mult	0,1,2

Options			
Gridsearch All	Yes	Gridsearch d only	No
Save Outcome	Yes	Show Warnings	No

Figure 4: Stress Test Inputs

## Conclusion

Maverik's initial need to forecast item-level candy bar sales can be satisfied by utilizing the predictive application created. Although potentially helpful, it was not practical to run predictions for all candy bar item numbers due to the amount of computing power required. There is a trade-off between utilizing enough historical data to produce an accurate forecast and the bottleneck of the machine's RAM to run those calculations.

Given the constraints of the packages compatible with Maverik's database and the size and scope of the data provided, we are confident the application created with the ARIMA algorithm will assist company management in determining the optimal item-level quantity of candy bars to stock in each store to maximize company profit. While some predictive models may be more general in nature, this application allows the user to customize the ARIMA model based on specific targets, allowing the degree of granularity and flexibility needed to address the initial business problem of forecasting specific item-level candy bar sales at each Maverik convenience store.

# Appendix

## Appendix A

### Item Stress test list

- min\_sales\_ids = [-44763,-30755,-30764,-40507,-48395,-40562,-40508,-34071,-35452,-48453]
- 1qt\_sales\_ids = [-40739,-41541,-42557,-32862,-45905,-46737,-44225,-37287,-45169,-35459] (quantities around 362)
- mean\_sales\_ids = [-11997,-41488,-31811,-41228,-38394,-44510,-12309,-46738,-40290,-11999] (quantities around 1609)
- 3qt\_sales\_ids = [-26802, -12248, -45240, -27618, -46720, -11963, -44540, -41259, -21206, -11968] (quantities around 4820)
- Max\_sales\_ids = [-12172,-29955,-21234,-12165,-12152,-12151,-12171,-12066,-16893,-14148]

## Appendix B

sample sales prediction output for item\_ID -14148 (MAR KG 3.29z SNICKERS 2 PIECE) for each store generated into CSV format for future reference

		2020-09-06	2020-09-07	2020-09-08	2020-09-09	2020-09-10	2020-09-11	2020-09-12
1	280,-14148	1.67	1.92	1.7	1.76	1.82	1.82	1.83
2	380,-14148	6.6	5.9	6.37	6.33	6.8	7.33	6.87
3	554,-14148	3.0	2.61	2.64	2.78	2.91	2.87	2.85
4	516,-14148	2.92	3.13	3.07	3.03	3.12	3.15	3.18
5	399,-14148	4.96	5.07	4.53	5.35	5.19	5.41	4.93
6	459,-14148	3.22	4.06	3.7	4.86	3.53	3.77	4.03
7	517,-14148	5.96	5.15	6.27	6.99	7.45	8.33	6.59
8	580,-14148	3.96	4.4	5.41	4.01	4.33	4.89	4.55
9	589,-14148	6.56	6.59	6.69	6.48	6.35	6.44	6.65
10	601,-14148	6.54	6.4	7.01	7.1	7.14	6.87	6.5
11								
12								

## Appendix C

sample error output log generated into CSV format for future reference

```
1 ,0
2 "(-44763, 280)",Combination Does not Exist
3 "(-30755, 280)",Combination Does not Exist
4 "(-40507, 280)",Combination Does not Exist
5 "(-48395, 280)",Combination Does not Exist
6 "(-40508, 280)",Combination Does not Exist
7 "(-34071, 280)",Combination Does not Exist
8 "(-48453, 280)",Combination Does not Exist
9
```

## Appendix D

Example Error Messages

