Predictive Analytics at Connecticut Children's Hospital

Major Qualifying Project
Sponsoring Agency: Connecticut Children's Medical Center
Advisor: Professor Sharon Johnson

Jacob Borges
Jihan Nabahani
John Tanny

5th April 2021

Abstract

Connecticut Children's Medical Center (CCMC) is a non-profit medical organization specializing in children's care. CCMC asked the MQP team to demonstrate the value of predictive analytics. The team developed a model that improved CCMC's forecasting accuracy for 88% of their metrics, an inventory demand model that expanded COVID testing, and an optimization model for behavioral health staffing that could save $1mil+/year. Adopting the team's models has highlighted the potential impact of further predictive modeling and industrial engineering work at CCMC.

Acknowledgments

We, Jacob Borges, Jihan Nabahani, and John Tanny, would like to express our deep and sincere gratitude to the many people who helped us throughout this project. First, a special thanks to our advisor Dr. Sharon Johnson, Operations and Industrial Engineering professor at Worcester Polytechnic Institute, for giving us invaluable guidance throughout the project. Her vision, motivation, and sincerity have deeply inspired all of us. It was a great privilege and honor to work and study under her guidance for seven months. We would also like to express our gratitude to Connecticut Children's Medical Center (CCMC) Chief Quality and Patient Safety Officer Dr. Lori Pelletier for sponsoring our project. Her guidance on project objectives and her feedback was crucial for the project's success. We want to extend our gratitude to the Continuous Improvement team at CCMC, especially Jessica Pereira and Deborah Berns. Their added insights and ability to gather and explain data and terminology were critical when developing our deliverables. We would also like to express our gratitude to the Enterprise Analytics Team, especially Kyle Lee, Renee Blanchette, Steven Burrows, Taylor Somma, Eric Dudas, and Kim Pier. Their ability to help us turn the capabilities we created into assets for the hospital was key to ensuring project success. Finally, we would like to express our gratitude to the entire medical staff, IT team, and all other staff members who work for such a great organization dedicated to improving children's physical and emotional health.

Table of Contents

Contents

List of Tables

List of Figures

1.0 Introduction

Connecticut Children's Medical Center (CCMC) is a non-profit medical organization located in Hartford, Connecticut, specializing in children's care. Its mission is to improve access to children's healthcare (Connecticut Children's, 2021). CCMC has various locations scattered around Connecticut, western Massachusetts and eastern New York. The Hartford location is the core of the CCMC health network, with typical hospital operations such as Inpatient, Outpatient, Surgery, and the Emergency Department. They also have a specialty group known as Connecticut Children's Specialty Group (CCSG), where patients meet a specific physician.

Recently, the COVID-19 pandemic had significant impacts on all healthcare organizations. CCMC has not escaped that. Operating under the many implications of the pandemic has provided various new challenges for the hospital. The project sponsor's overall goal was to use this project to highlight the need for and the importance of predictive forecasting and industrial engineering work at CCMC. Several specific areas of focus were identified for the project.

The first area of focus was forecasting. The unprecedented fluctuations of revenue during the COVID pandemic motivated the organization to use demand predictions when making senior-level decisions. The pandemic acted as a catalyst for developing a more robust forecasting system for predicting patient volumes. The team developed an automated forecasting system that improved the accuracy of the current system.

The hospital's operations and resources were also significantly impacted by the pandemic. Infection prevention measures strained staffing, space, and personal protective equipment (PPE) within the hospital. However, if the patient tests negative for Coronavirus, Flu, and RSV, the patient can be treated without such prevention measures. A new rapid test called the "Lab in a Tube" (LIAT) test was a critical new asset for CCMC to streamline this testing process, but they only had a specific government rationed supply of these tests. By developing an inventory model for the LIAT tests, the hospital could maximize the number of patients who could take a LIAT test without risking a stock out. Better yet, patients who took a LIAT test received results within 20 minutes, significantly improving the speed and quality of care. The

LIAT inventory model helped aid the decision-making process for the management of these LIAT tests.

Finally, the behavioral health unit has been continually growing for several years, and the pandemics' impacts have caused additional challenges. There is limited staffing between patient care partners (PCPs), patient care assistants (PCAs), and registered nurses (RNs). Figuring out how to optimally staff the complex behavioral health units was challenging as it was unclear whether the shift in demand was temporary or permanent. Leadership and hiring managers need to determine the optimal staff levels, and operations staff must know how to serve the patients best given their current staffing. Both of these challenges are handled with limited analytical support. The team's objective was to create an optimization model that could help identify challenges, compare scenarios, and propose optimal behavioral health staffing solutions.

Throughout the project, the team was able to work with experts from two different departments to tackle these objectives. One team was the Continuous Improvement team, which served as the bridge between analytics and operations. Several experts on this team aided the project by gathering data, answering questions, and providing critical insight to project objectives and methodologies to ensure deliverables fit well with the framework of the current operations at CCMC. The other team was the Enterprise Analytics team composed of analysts and computer scientists who use data and modeling to develop analytical strategies and business decisions. This team was instrumental in getting access to data sources to fuel analytical solutions and vetting analytical tools and processes to ensure they fit within the enterprise system at CCMC. Both teams were highly attentive to the project goals and provided immense support in shaping the project deliverables.

2.0 Background

The following section describes the COVID-19 virus and how it presented operational challenges to medical facilities, forcing the facilities to forecast its surge. Additionally, there is background information on modeling variability in hospital capacity and how it can lead to operational challenges. The third part of this section talks about hospital staffing during COVID-19 and how it has changed some staff's working patterns. After that, the focus shifts to current analytics and forecasting methods at CCMC, followed by descriptions of how the team researched the ARIMA model, its parameters, its implementation, and its limitations. Next, a section related to the background of the LIAT testing scenario and its challenges. Finally, the last section describes optimization models and the framework of the behavioral health staffing optimization problem. All of this background research and detail was critical to understanding and developing methodologies and results.

2.1 COVID-19

The novel COVID-19 virus that has caused the global 2020 Coronavirus Pandemic is a member of a larger body of viruses labeled ‘Coronaviruses.’ They are characterized by crown-like spikes on their surface. Coronaviruses have four main subgroups, known as Alpha, Beta, Gamma, and Delta. Human coronaviruses were first identified in the mid-1960s. The seven coronaviruses that can infect people are: Common human coronaviruses: 229E (alpha coronavirus), NL63 (alpha coronavirus), OC43 (beta coronavirus), HKU1 (beta coronavirus). Other human coronaviruses: MERS-CoV (the beta coronavirus that causes Middle East Respiratory Syndrome, or MERS), SARS-CoV (the beta coronavirus that causes severe acute respiratory syndrome, or SARS), SARS-CoV-2 (the beta novel coronavirus that causes coronavirus disease 2019, or COVID-19), (*Coronavirus*, 2020). The 2020 COVID pandemic acted as a catalyst for most of the work that occurred during this project. It forced the CCMC organization to scrutinize forecasting and resource efficiency due to the pandemic's new regulations and practices. Though this team did not perform any direct Coronavirus infection or associated death forecasting, it was essential to understand the pandemic's current and future impacts on operations and leadership decision-making in the team’s analysis. Without the Coronavirus pandemic, the project would most likely not have had the same motivation.

2.2 Modeling Variability in Hospital Capacity

Connecticut Children Medical Center and other medical facilities generally face challenges in determining bed capacities because of the variability in the number and type of patients who frequent those facilities each day. Determining required bed capacity gives the medical facilities the opportunity to plan their staffing requirements resulting in efficient operational processes and a higher medical service level. This section describes a peer-reviewed journal article by Mackay (2005) evaluating a stochastic version of the Harrison—Millard simulation model that was developed using 1-year data of the year 1999 from a hospital in Adelaide, Australia, to understand the variability in demand in the number of patients for both elective and emergency visits. A Poisson distribution was used for patients' admissions since independent events are occurring within intervals of time. A chi-squared test rejects this hypothesis because there is a significant difference in the number of patients' arrival by different days of the week.

Another effect presented in the model is seasonality because hospital admission rates increase dramatically during a flu season compared to any other time of the year. Therefore, the best possible way to model this was to run each day of the week separately (Monday-Sunday). The model faced a challenge regarding the discharge process because the patient's length of stay depends on the patient's condition and the stages/ departments that he has to go through. The researchers found that it is best to use exponential distribution to fit the occupancy times, as shown in Table 1(Mackay, 2005).

Table 1: Occupancy Times in Hospital Mackay Example

	Current length of stay, i.e. days in hospital since admission								
Date	0	1	2	3	4	5	6	7	8
1/1/99	22	22	24	6	8	8	4	4	2
1/2/99	**28**	21	21	21	6	5	6	4	3
1/3/99	18	**26**	18	17	21	6	2	6	4
1/4/99	22	17	**26**	17	15	21	6	2	6
1/5/99	30	18	14	**21**	9	13	16	4	1
1/6/99	29	28	13	8	**16**	6	10	12	4
1/7/99	31	28	23	10	6	**15**	6	4	12
1/8/99	32	30	23	19	8	6	**14**	3	3

The rows represent specific dates of the week, whereas the columns represent the length of stay. The body of the table shows the number of bed occupants for that day for each length of stay. Most models assume constant admissions on different days of the week, which is not the case in real life.

The paper also discussed the effect of the size of patient volumes. For example, suppose the number of COVID cases increases. This will affect the mean arrival of each day of the week but will keep some departments' discharge process the same. The model allowed the exploration of variability in the arrival rate and time spent at the hospital and prepared the hospital to allocate more resources were needed as demand grows during a specific time of the year; thus, increasing the efficiency of hospitals operations. The ability to understand and model variability as described in this research was a critical aspect of the project, especially due to the increased variability that impacts of the Coronavirus pandemic brought.

2.3 Hospital Staffing During COVID-19

One of the challenges that Coronavirus poses to healthcare is staffing availability. Some staff may get infected with the virus. Others may have to self-isolate themselves due to an infected family member. The unavailability of some staff may lead to several challenges in the hospital's operations. A group of doctors in England developed a model to simulate staffing status under different infection rates. (Schooling et al., 2020). The model considers the number of staff in a hospital, their working pattern, and the number of areas they need to cover. Infection rate percentage can be entered daily. The graphs below represent the percentage of staff at risk of

not being available during a specific day for two different departments. As shown in Figure 1, there is an extra daily risk percentage for being off work due to infection in various hospital departments.

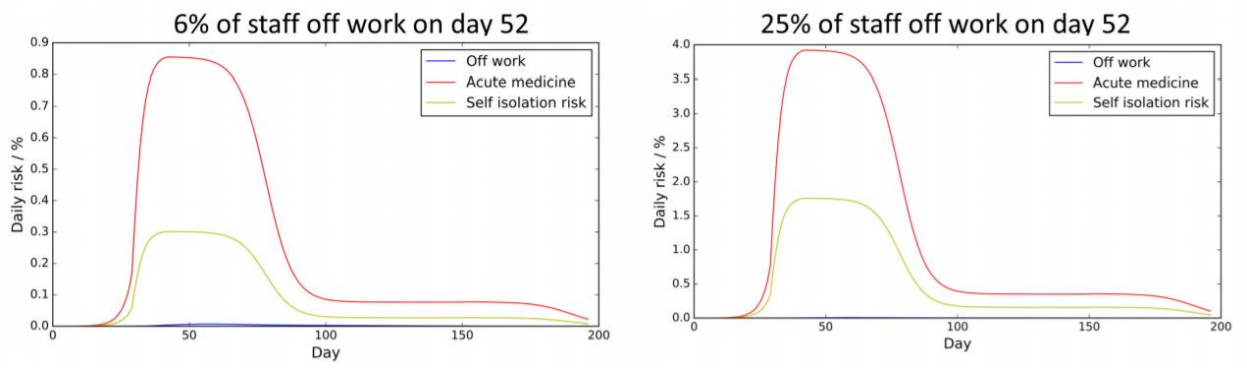


Figure 1: Graphs for percentage of staff at risk based on model created by doctors in England (Schooling et al, 2020)

Variability in staffing was particularly critical to understand the behavioral health staffing optimization problem of this project. Variability brought on by the pandemic only increased impacts in staffing availability and high turnover rates in critical positions at the hospital. Accounting for this variability and understanding how it affects staffing was a core aspect to analyzing the behavioral health staffing challenge.

2.4 Analytics and Forecasting at CCMC

The Enterprise Analytics team at CCMC has been collecting data on ten metrics. These metrics provide an overview of the hospital's operations. Table 2 presents these metrics and their associated descriptions. The reader is advised to reference this table when needed, as we will use these metrics without explanation.

Table 2: Current Volumes at CCMC with Descriptions

Volume	Description

IP Census Days	Any inpatients who are admitted or discharged
NICU, PICU & MS 8 IP Census Days	Number of patients who are newborns and ones who receive pediatric intensive care
All Other IP Census Days	Sum of inpatient service days
Total Discharges	Number of patients leaving the hospital after receiving care
ED Visits	Number of patients who receive care in the emergency department
Clinical Support Appointments	Number of appointments scheduled with a physician or with other health professional
CCSG Appointments	Number of appointments scheduled with the specialized group at CC
Total Appointments	Total number of appointments during the given period
Surgeries	Number of surgeries scheduled
Behavioral Health	Number of patients admitted as behavioral health patients

CCMC's management team has been forecasting these metrics since April 2020, 2 months after the first spike in COVID-19 cases in March. The impacts of the Coronavirus brought new attention to the importance of sound, data-based decision making. In an effort to prepare the hospital for a resurgence in the COVID-19 virus, CCMC identified 3 different scenarios: Slow (0-3 COVID patients), moderate (3-9 COVID patients), and fast (9-24 COVID patients), each with their own impacts on expected patient volumes. Some outputs of these forecasts can be seen in figure 1 below.

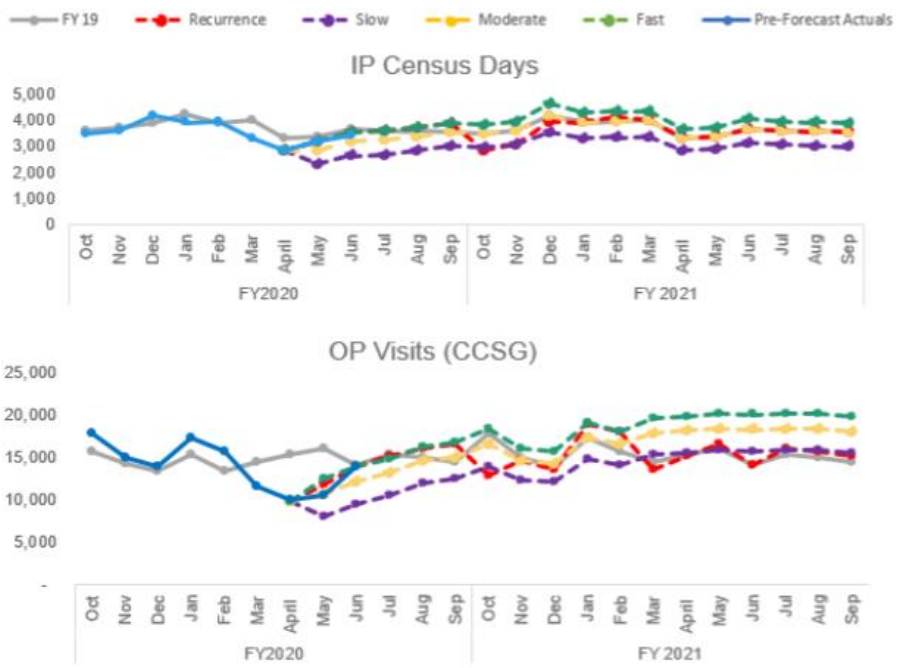


Figure 2: Current Modelling Outputs at CCMC

These graphs show the expected impact on patient volumes based on the CCMC Analytics team's manual forecasts. To generate these forecasts, an analyst takes the following steps every month:

1. Manually queries and saves data for metrics of interest
2. Takes 2 standard deviations of last year's data to generate the forecast's upper and lower bounds.
3. Manually estimates the forecast for each scenario, attempting to match trends in the data
4. Manually adjusts the forecast based on market research data

CCMC's current forecasting system requires up to 16 hours a week to complete steps 1-3. Therefore, this project's forecasting objectives were to create a robust forecasting methodology that could predict these metrics with greater accuracy and less manual work. This would allow analysts to spend more time making educating decision-makers and less time creating the forecasts.

2.5 ARIMA Forecasting Method

ARIMA stands for Autoregressive Integrated Moving Average. For data to be fit with an ARIMA forecast, it must meet three assumptions. First of all, the data is cyclical in nature. The data should have some trend that is dependent on time which is different from seasonality. For example, if every month, there is a slight increase in ED visits, and then it drops off. Secondly, there is a dependency that exists between observations and the residual error. For example, if tomorrow there is 500 times the number of ED visits as there were today. The error would also increase by an amount that is relatively proportional to the quick increase in ED visits. Thirdly, there is a dependency between observation and some number of lagged observations. For example, ED visits volume for today is dependent on ED visits in the past. In other words, "The ARIMA model is a filter that tries to separate the signal, such as some consistent variation in ED visits or some trend in the data, from random error known as noise. The signal is then taken and extrapolated in the future into a forecast" (Michael Kane, 2020).

However, what happens when data does not meet all three of these assumptions? The beauty of ARIMA is that when implemented appropriately if data does not meet one of these assumptions, the model simplifies itself to the more appropriate model. For instance, a plain Moving Average may fit better than a full ARIMA model for some data. In that case, the model simply adapts to fit the simpler form.

To implement ARIMA, there are three parameters to consider: P, D, and Q. Each parameter relates to the three assumptions mentioned previously. The first of these variables, P, is representative of the number of lagged observations. For ED visits, for instance, P is determined by graphing the partial autocorrelation of ED visits and checking how many data points exist outside of the 95% confidence interval. The graph below shows 1 data point outside the confidence interval of ED visits data set: therefore p=1.

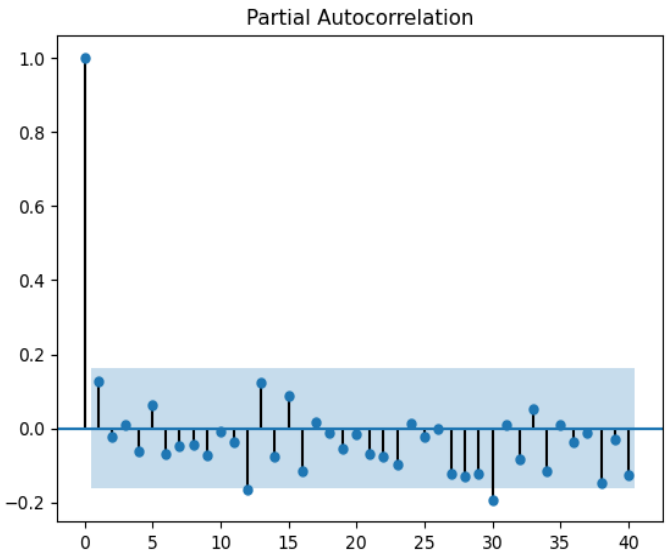


Figure 3: ARIMA Parameter P Calculation Graph for ED Visits

Second, D is equal to the number of times the row observations are different to make the data stationary. D is determined for ED visits and other volumes by graphing the autocorrelation using first-order and second-order differencing. The first order differencing of ED visits shows 1 data point outside the 95% confidence interval in Figure 4. The second-order differencing shows 2 data points outside the 95% confidence interval; therefore, the data is differenced 1 time only. Another way to find d is using the ADF Test for Stationary Data. If the test results in p-value > 0.01, the data must be differenced.

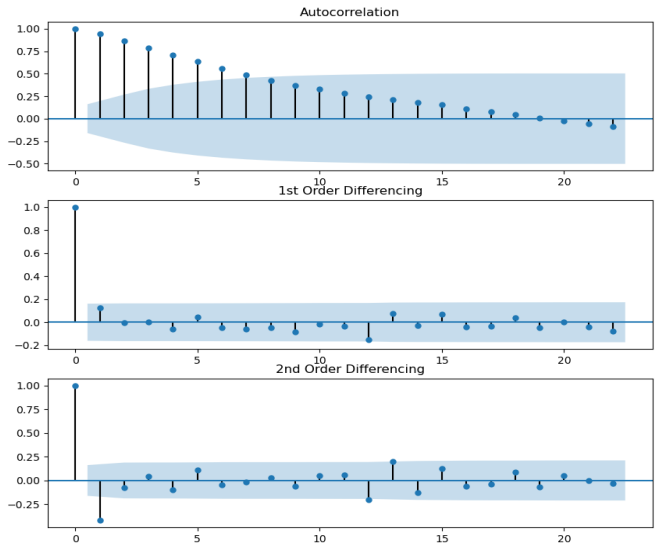


Figure 4: ARIMA Parameter D Calculation Graphs for ED Visits

Finally, q is the number of lagged forecast errors. For q, graph the autocorrelation with no differencing. The graph of ED visits below shows 1 data point outside the confidence interval for lagged errors: therefore q=1.

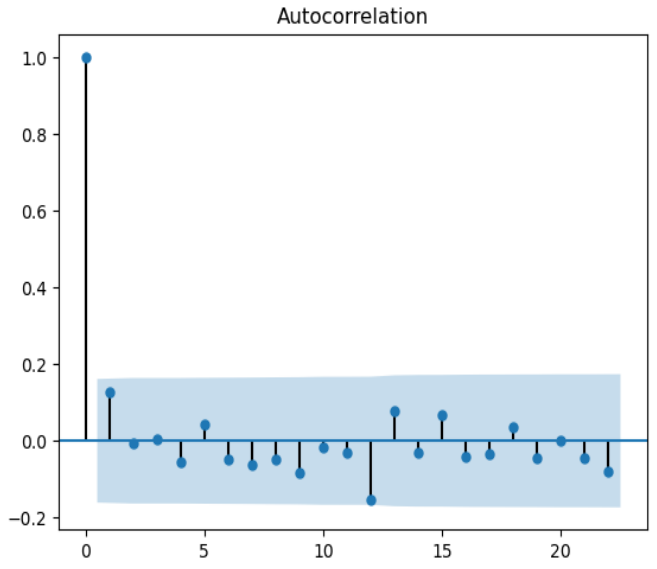


Figure 5: ARIMA Parameter Q Calculation Graph for ED Visits

Therefore, for ED visits, the parameters are (1,1,1).

Each instance of the ARIMA model puts out an ARIMA formula used for forecasting as well. For ED visits of parameter (1,1,1), ARIMA produced the following forecast formula:

$$\hat{Y}(t) = \mu + Y(t-1) + \phi(Y(t-1) - Y(t-2)) - \theta e(t-1)$$

where Phi is the autoregressive coefficient, theta is the lagged forecast error, and e(t-1) is the error at period t-1. The formula is changed automatically as the parameter of the volume changes. The other volumes' parameters can be found in accuracy table 4 using the ARIMA tool at the results section.

Even though ARIMA is a suitable forecasting technique where trend and variation exist, it does have some limitations. One of them is the fact that CCMC has only a few years of data, whereas more data could be more valuable for the ARIMA model. Another limitation of ARIMA is implementing it correctly can require large amounts of computer resources to train the model and identify the best P, D, and Q values. Lastly, while the model can be effective in capturing seasonality and the overall trend, it can fall short in forecasting values that fall significantly outside the norm.

2.6 LIAT Testing

Due to the pandemic's impacts, CCMC must administer tests to four categories of patients for COVID, the Flu, and RSV. These categories are admitted patients (a patient who needs to stay overnight), behavioral health patients, urgent surgeries, and transfers from other hospitals or CCSG clinics. Until these tests come back negative, these patients are treated as if they have one of the three viruses, which puts a strain on staffing, space, delay in care and Personal Protective Equipment (PPE) requirements.

A new development that has benefited CCMC testing capacity is the “lab in a tube” (LIAT) testing capabilities. These tests can be run through analyzers that return results in 20 minutes. CCMC has two of these analyzers and each week they are supplied with 200 LIAT test kits that they can stockpile if not used. By applying the LIAT tests, CCMC can get results quickly and therefore care for patients very quickly without having to hold them in containment

areas for up to a day while they wait for test results from another lab which would take 12-60 hours from sample collection.

With this new capability CCMC needed to understand what their usage of the tests would look like so that they could properly plan for testing of the key patient groups. They needed a model that would allow them to understand what their current inventory for LIAT tests was, what the expected supply was going to be, what the patient demand was going to be, and what the long-term inventory burndown for demand would look like. A model that gave them this information would allow the LIAT planning team to be able to determine what patient groups could and should be eligible for these tests. The four patient groups listed above were the primary focus, but were the volumes of those patient groups going to be above or below the supply volume? If there was not enough supply, they needed to know as soon as possible so they could restrict the patient groups even more to only those who critically need a test. If there was excess supply, it would give the hospital an opportunity to expand the testing to other patient groups and improve their patient experience as well. A forecasting model was required to answer these questions.

2.7 Optimization Problems

Optimization problems are ideal for determining the best solution to a problem based on given constraints. As is described by Boyd and Vandenberghe, “the notation [for an optimization problem is to] minimize f0(x) subject to fi(x) ≤ 0, i = 1, . . . , m hi(x) = 0, i = 1, . . . , p to describe the problem of finding an x that minimizes f0(x) among all x that satisfy the conditions fi(x) ≤ 0, i = 1, . . . , m, and hi(x) = 0, i = 1, . . . , p (Boyd et al, 2004). The key characteristics of an optimization problem are an objective function, decision variables, and constraints. The objective function is an equation representing the outcome of the problem and is the target for optimization. This objective can either be maximized or minimized. Optimization problems are ideal for determining the best solution to a problem based on given constraints. As is described by Boyd and Vandenberghe, “the notation [for an optimization problem is to] minimize f0(x) subject to fi(x) ≤ 0, i = 1, . . . , m hi(x) = 0, i = 1, . . . , p to describe the problem of finding an x that minimizes f0(x) among all x that satisfy the conditions fi(x) ≤ 0, i = 1, . . . , m, and hi(x) = 0, i = 1, . . . , p (Boyd et al, 2004). The key characteristics of an optimization problem are an

objective function, decision variables, and constraints. The objective function is an equation that represents the outcome of the problem and is the target for optimization. This objective can either be maximized, minimized, or set to achieve a targeted value. The decision variables represent the variables of the optimization problem that can be changed and altered to affect the objective function. This changing of the decision variables is the action taken to attempt to meet the optimized form of the objective function. Constraints are also common assets of an objective function. They specify certain rules about the optimization problem. The domain of the optimization function is representative of the set of points for which the objective and all constraints are defined. Optimization problems can be solved by a variety of different software or through manual manipulation.

Excel Solver is a common software used for solving optimization problems. It is a free add-in to Microsoft Excel. The Excel Solver engine takes in all the variables listed above. The user can set an objective function, decision variables, and constraints described in the "Define and solve a problem using Solver" pages of the Microsoft support website (Define and solve a problem by using solver, n.d.). The Solver engine choices are Simplex LP, GRG Nonlinear, and Evolutionary. Simplex LP is built to solve problems that are linear, GRG nonlinear is built to solve problems that are smooth and nonlinear, and Evolutionary is built to solve non-smooth problems. The linearity and smoothness of a particular problem are defined by the degree of the objective function and its constraints. Solver uses these engines to cycle through permutations of the decision variables in order to optimize the objective function within the domain of the model.

2.8 Behavioral Health Staffing Optimization

Within CCMC, care for behavioral health patients is becoming an increasingly critical area of focus. Volumes for behavioral health patients continue to grow rapidly within the hospital, and staffing is becoming increasingly difficult. The Coronavirus pandemic has further exacerbated this. One of the key elements driving this challenge for the behavioral health units is patient care partners (PCPs). PCPs are staff members at CCMC whose specific role is to sit with behavioral health patients who have a certain level of acuity and require a constant monitor. PCP requirements per patient vary based on the patient's acuity. Some patients require 2 PCPs, while other patients may share a PCP due to having only specific time-oriented needs. The PCP

profession can be quite mentally taxing, and the turnover rate is relatively high. As a result, it can be challenging for the hospital to maintain the required number of PCPs staffed to meet the continually growing behavioral health needs.

To meet these growing needs, other staff within the hospital are often required to cover these PCP roles. The most common candidates for this are patient care assistants (PCAs) and registered nurses (RNs), who have other duties they need to fulfill. They are also more expensive than a PCP, so having them do PCP work is not an efficient use of hospital funds. Overall, this phenomenon has led to challenges in meeting bedside coverage goals, represented as the percentage of behavioral health patients receiving the PCP care they require. This issue presented a great opportunity for optimization methodologies to answer two key questions:

1) "What is the optimal staffing required to meet behavioral health patient sitting needs across all shifts?"

2) "What is the minimum shortage of patient sitters that can be optimally achieved given the current staffing model?"

The audience for question 1 is leadership and management who are responsible for hiring and managing behavioral health staff. Question 2's audience is the operations floor who are responsible for ensuring the highest bedside coverage based on the current available resources.

At CCMC, the staffing schedule is split into six time blocks of four hours each starting at 0300 hours. Staffing is also split amongst five key departments which are the ED, MS6, MS7, MS8, and PICU departments. The models will both aim to answer the above listed questions across the time blocks and departments listed in a way that is dynamic to adapt to hospital operations changes and needs.

3.0 Methodology

The following section describes the methodologies the team used to produce the deliverables of the project. First, the ARIMA methodology was selected as a best practice for predictive modeling of 10 key hospital variables. Additionally, a section is dedicated to creating the LIAT demand model using excel inventory modeling techniques. The final subsection reports on the optimization methodology of excel solver and how it was used to develop models to optimize behavioral health staffing resources. Each methodology was critical to developing the project's deliverables in a way that was repeatable and achieved best practices of data analytics.

3.1 ARIMA Model

To construct the forecast model, we first had to assess the problem and investigate the CCMC's current forecasting methods. After looking at 3 years of data for each metric, we found that the data was generally stable (excluding COVID-19 timeframe), with a fair amount of variation that seemed to be unexpected, making it a good candidate for ARIMA. Similarly, the current forecasting process was time consuming, error prone, and did not utilize statistics.

We chose to use ARIMA as our intended forecasting technique. It was bound in statistics yet was flexible enough to conform to any data we provided it. The benefits of other simpler forecasting techniques, like an exponential smoothing model, are captured in ARIMA. Better yet, there is an open-source ARIMA library in Python 3, making it straightforward to implement.

Once we identified that we would use ARIMA as our forecasting technique, we needed to learn ARIMA in Python 3. We made our first ARIMA model using sample data, following the guidance of Jason Brownlee, a professional Machine Learning developer with a PhD in Artificial Intelligence (Brownlee, 2020). Once we were confident that ARIMA would work for our intended purposes, we created the ARIMA model for CCMC. The model is adaptive as it can produce a forecast of any length with any time series data input. The model also implements proper machine learning principles like hierarchical cross validation to prevent overfitting. Once the model was developed, we ran it on the CCMC data. To ensure that the model was working well, we compared the model's error to CCMC's manual forecast error. Generally, across all 10 metrics we had a more accurate model.

With an accurate and successful model, it was time to implement it. We worked with two data engineers at CCMC to develop a process that would gather the data, create the forecasts, and distribute them to the analyst. An ETL server would best suit the model's needs. After conversations with the analyst, we decided upon the final format for the forecasts and a distribution method.

3.2 LIAT Inventory Model

The questions that CCMC sought to answer for this project were how many of the LIAT tests to use and when. To answer these questions, the team used excel modeling, modifying historical data for COVID impact, and including variation based on standard deviation. The team identified the patient groups that were candidates for the LIAT tests with help from the experts at CCMC running the LIAT project. Then, historical data was pulled for these patient groups. This historical data is very cyclical in nature, so forecasts were based on the previous year's data with modifying values based on market conditions. The modified percentages to be applied to each of the patient groups was gathered from current volumes compared to past years volumes and expert opinions from leading doctors and data analysts at CCMC.

Once modifiers were applied to the historical data, it yielded a predicted forecast for LIAT usage if the selected patient groups were given tests. However, variation has a significant role in this model. To account for this, the team also developed an upper and lower bound forecast, representative of two standard deviations above and below the mean forecast. This standard deviation was calculated based on the sum of squared errors between the actual volume of patients and the predicted volume. The net result was three weekly forecast lines showing the predicted demand and upper and lower bounds of that predicted demand for LIAT usage.

The team also worked with the supply chain team to gather data on the current volume of LIAT tests as well as the supply rate. The current volume at the start of LIAT usage was 2560 LIAT tests with a supply receive rate of 200 tests per week by government ration. This data was the final piece of the puzzle that was used to create a complete inventory model. The team was able to compare the predicted demand for LIAT with the supply arrival to predict what the inventory for the LIAT tests will be for as long as the supply remains at the 200 per week value.

Predicting out through the month of February 2021 was the first key target so that Connecticut Children's management teams could understand if they needed to be more conservative with the tests or if they could open use of them to different patient groups. Further forecasting occurred beyond that point using the same methodology and adjusting the inputs based on additional patient groups and changing supply.

3.3 Behavioral Health Staffing Optimization Model

The two main questions that CCMC needed answering for this sub-project were "What is the optimal staffing required to meet behavioral health patient sitting requirements across all shifts?" and "What is the minimum shortage of patient sitters that can be optimally achieved given the current staffing model?" The chosen methodology to answer these questions was using Excel Solver optimization models. Each of these questions required its own model, but the models shared many input sources and assumptions. The first key input source that had to be gathered with the help of the continuous improvement expert for behavioral health was the average requirements for behavioral health patient care partners by shift and by department. As was described in the background, the hospital shifts are set up in 4-hour time blocks, and there are 5 key departments to focus on. A spreadsheet that is updated daily by the charge RN for each department and shift lists the PCP historical requirements and serves as the first key input. The data from this spreadsheet is organized and displayed in a dynamic pivot table. The rows represent average PCP requirements per time block, whereas the columns represent the department. This input data is then pulled into each of the models.

Month	(All)					
Grid Type	(All)					
Average of 1:1 Sits Required	**Column Labels**					
Row Labels	**ED**	**MS6**	**MS7**	**MS8**	**PICU**	**Grand Total**
0300-0700	3.983108108	3.044701987	1.870431894	1.102040816	0.4	2.097090663
0700-1100	4.124590164	4.849673203	1.898360656	1.118243243	0.425087108	2.517678452
1100-1500	4.113861386	4.849673203	1.861386139	0.993197279	0.403508772	2.482562039
1500-1900	4.240924092	4.866449511	1.92384106	0.996621622	0.409961686	2.5609258
1900-2300	4.085808581	4.68627451	1.8125	1.067567568	0.363636364	2.467752885
2300-0300	4.052459016	3.050324675	1.862745098	1.101351351	0.393728223	2.120173103
Grand Total	**4.100550964**	**4.226158038**	**1.871499176**	**1.063205418**	**0.399640503**	**2.373934022**

Figure 6: Average PCP Requirements in each Department

Model 1: What is the optimal staffing required to meet behavioral health patient sitting needs across all shifts?

Along with the inputs for the sitter requirements pivot table, this model takes inputs of costs per shift for PCPs, PCAs, and RNs. These values were determined using job descriptions from CCMC. Inputs for the maximum achievable number of hires for each staffing type by shift are also included. These are critical due to the high turnover of PCPs and the difficulty of hiring them at the quantity the hospital may need. The final inputs for this model are the bedside coverage targets by time block. Currently, those targets are set to 100%. All of these inputs are dynamic in the model and can be changed at any time to either run different scenarios or match impacts of change within the hospital.

Model 2: What is the minimum shortage of patient sitters that can be optimally achieved given the current staffing model?

Like the first model, this model also pulls the sitter requirement inputs from the pivot table. In addition to that, inputs for the current available staffing by time block are required. The decision variables for this model are the staffing allocations by time block and by department. These decision variables of staffing allocations then have the current staffing availability by their respective department and time block subtracted from them. This yields a value representing the number of PCPs short during that time block and in that department in order to meet bedside coverage needs.

Both of these models run on the Simplex LP solver engine through the free excel solver add-in. One of the key aspects of the models that were critical to this methodology is the ability to alter inputs and/or change data sources and rerun the model with ease. The hospital environment, especially in the behavioral health floors, is constantly changing, and these modeling methodologies need to be able to adapt to those changes. For this reason, all inputs are dynamic and do not require any changing of the solver model when altered. The models can simply be rerun with the new inputs and will give the new solution. Additionally, it is important to note that both models do have the possibility of infeasibility. This is a reasonable output for either of them, and it indicates that with the current loaded staffing and/or bedside coverage

goals, the constraints cannot be met. Therefore either staffing resources should be raised, or bedside coverage goals should be lowered.

4.0 Results

This section discusses the outcomes of the ARIMA Forecast, LIAT Testing Model, and the Staffing Optimization Model. In short, the ARIMA model proved to be accurate and dependable when we tested it with a wide range of data over different time periods, outperforming CCMC's manual forecasts in 90% of the tested metrics. The LIAT Testing Model successfully predicted the inventory of the LIAT tests, increasing the speed and quality of care at CCMC. The Behavioral Health Staffing Optimization Model was able to determine the cost impacts and optimal staffing allocations in order to meet bedside coverage goals or behavioral health patients.

4.1 ARIMA Results

The ARIMA model that the team developed produces a forecast each time the model is run for a variable. This section of the paper will share how and why the model was developed, including sample code. The length of the forecast, i.e., the number of time units that the model predicts a value, is specified by the end-user of the model. The end-user can even request that the model makes multiple forecasts, each of different lengths if they wish to do so.

The ARIMA Model was tested on the ten metrics that CCMC forecasts. The results of the ARIMA forecasting can be summarized in Table 3. The parameters of the volumes were calculated automatically by testing every combination for the lowest amount of error on the most recent 20% of data. For instance, for ED visits, the best optimal parameter was (1,1,1), since these parameters produced a forecast that minimized the error over the most recent 20% of data available. The accuracy was determined by calculating the Normalized Root Mean Squared Error (NRMSE). For more information, see Figure 9. For instance, ED Visits had 5%, 6%, and 14% NRMSE corresponding to 4, 6, and 12 weeks of forecasting, respectively. As shown in Table 3, the NRMSE traditionally increases over time.

Table 3: ARIMA NRMSE Extended Forecasts and Parameters

Volumes	Parameters	4 Week Forecast	6 Week Forecast	12 Week Forecast
IP Census Days	10,1,1	4%	5%	7%
NICU, PICU & MS 8 IP Census Days	0,0,1	6%	6%	6%
All Other IP Census Days	8,1,2	1%	10%	15%
Total Discharges	0,2,1	11%	9%	13%
ED Visits	1,1,1	5%	6%	14%
Clinical Support Appointments	6,0,1	9%	10%	15%
CCSG Appointments	4,1,2	9%	11%	14%
Total Appts	6,2,0	10%	9%	14%
Surgeries	6,0,0	18%	19%	22%
Behavioural Health		18%	15%	18%

To ensure that the ARIMA Model would have a positive impact on CCMC's operations, it was tested against CCMC's manual forecasting technique, covered in section 2.4. Table 4 compares the accuracy of CCMC's forecast and the ARIMA model. The cells highlighted in green show a smaller NRMSE compared to CCMC. The cells highlighted in yellow represent a margin of error similar to CCMC NRMSE. However, the red cell shows that the ARIMA model was less accurate than CCMC for the Surgeries category. Overall, the ARIMA NRMSE was lower than 90% of Connecticut Children's forecast method's NRMSEs.

Table 4: ARIMA NRMSE Compared to Current State at CC. January 2021.

	CC Prediction Accuracy Update (just July 2020 and August 2020)	ARIMA NRMSE, 3 Week Forecast, Great Data Quality
IP Census Days	9%	5%
NICU, PICU & MS 8 IP Census Da	9%	6%
All Other IP Census Days	14%	11%
Total Discharges	12%	8%
ED Visits	13%	7%
Clinical Support Appointments	20%	14%
CCSG Appointments	15%	14%
Total Appts	16%	14%
Surgeries	15%	20%
Smaller RMSE	Relatively similar NRMSE	Greater NRMSE

The rest of section 4.1 will cover the basic functionality of the ARIMA model. For demonstration purposes, the model was set to produce forecasts for 4, 6, and 12 weeks for the IP Census Days data point. Firstly, when an end-user has identified data that they would like to forecast, it is important to get a feel for the data. Graphing the data and exploring its characteristics is necessary. Figure 7 explores the data visually and prints out the date of each data point.

```
# look at data
print(series.head())
date = series.index
print(date)
datelist = []
datelist.append(date.strftime('%m/%d/%Y').tolist())
print(datelist[0])
datelist1 = datelist[0]
series.plot()
plt.show() # if this plt has a clear trend, we know it will need some
differencing of at least one
```

Figure 7: Look at the data: Grid Search Arima Parameterization.py

Now that the data is verified, it is time to identify the best P, D, and Q values for the ARIMA model to fit the data. As explored in section 2.4, those values can be manually identified. Manually identifying the P, D, and Q values is an important exercise. It teaches the user a lot about the data and how ARIMA will make its forecasts. However, it is tedious and manual. To speed up and improve the accuracy of the process, we have developed the *Grid Search Arima Parameterization.py* script that automatically identifies the best P, D, and Q values using a technique known as grid searching.

This evaluate_models function in Figure 8 from the *Grid Search Arima Parameterization.py* script takes a range of P, D, and Q values and a dataset as an input. The function will test each different combination of the P, D, and Q values. Whichever combination best fits the dataset will be returned as an output of this function.

```
def evaluate_models(dataset, p_values, d_values, q_values):
      dataset = dataset.astype('float32')
      best_score, best_cfg = float("inf"), None
      for p in p_values:
            for d in d_values:
                  for q in q_values:
                        order = (p,d,q)
                        try:
                              mse = evaluate_arima_model(dataset, order)
                              if mse < best_score:
                                    best_score, best_cfg = mse, order
                              print('ARIMA%s MSE=%.3f' % (order,mse))
                        except:
                              continue
      print('Best ARIMA%s MSE=%.3f' % (best_cfg, best_score))
      return(best_cfg)
```

Figure 8:evaluate_models function: Grid Search Arima Parameterization.py

Calculating the error of the model is an important part of testing this model, identifying the best P, D, and Q values, as well as trusting the model's forecast. The model's error is calculated using an error calculation known as the Normalized Root Mean Squared Error (NRMSE). This calculation takes the square root of the mean squared error of the actual versus predicted values of the testing dataset and then normalizes that value with the mean of the whole dataset.

```
error = (sqrt(mean_squared_error(test1, predictions1)))/(s.mean(X))
```

Figure 9: Normalized Mean Squared Error formula:

Once the user has explored the data, found the best P, D, and Q values, and the error calculation, it is time to evaluate the ARIMA model on the current data and develop a forecast. To evaluate the ARIMA model, the data is first split into a training and a testing set. The training set of data ARIMA will use as historical data. The model will fit the ARIMA model specified by P, D, and Q to this historical training data. The model selects the oldest 80% of the input data as the training data.

The testing set of data allows the user to evaluate how well the model's forecasts will have performed if it were making forecasts over that time period. The testing set of data is the

most recent 20% of the input data. The model uses a for loop to cycle through the test set of data. For the sake of example, assume a forecasting length of 4 weeks, where weeks is the unit of time. First, the model is defined using the P, D, and Q parameters, and then it is fitted to the training set of data. Next, a 4-week forecast is created. In the next iteration of the for loop, the model is again fit to the training set of data, plus the first four weeks of the testing set of data. A 4-week forecast is created again, and the for loop repeats in this fashion till the end of the training set. See the evaluate models function in Figure 10.

```
forecast = z # set forecast equal to the forecast week length

## Using a rolling model for predictions, with the rolling period =
forecast length. This method will record the error of ARIMA over the
testing period. All variables unique to this section are denoted with a 1
      history1 = [x for x in train1] # create history list. This will
store the historical values (rolling) during the for loop.
      #print(type(history))
      predictions1 = list()
      #print(len(test)//forecast) # will always round down

      for t1 in range(len(test1)//forecast): # for loop for 1/forecast of
the weeks in test, since we are testing a forecast of length forecast
            for l1 in range(forecast-1):
                  if len(test1)/forecast != len(test1)//forecast:
                        test1.pop() # removes the last occurance of test
                        #print(test)
            if t1==0:
                  model1 = ARIMA(train1, order=orderlist)
                  model_fit1 = model1.fit(disp=0)
                  output1 = model_fit1.forecast(forecast)
                  #print("output1: ")
                  #print(output1)
                  yhat1 = output1[0] # get the guesses for the next
forecast weeks
                  yhat1 = yhat1.tolist() # make guesses into a list
                  #print("yhat1: ")
                  #print(yhat1)
                  predictions1.extend(yhat1) # add forecasted values into
predictions list
                  #print("predictions1: ")
                  #print(predictions1)
                  #print("test")
                  #print(test)
                  obs1 = test1[t1:t1+forecast:1] # get the next actual
values from test
                  #print("obs1")
                  #print(obs1)
                  history1.extend(obs1) # add to history with obs
                  #print("history1")
                  #print(history1)
            else:
                  #print("predictions1 :")
```

```
                #print(predictions1)
                model1 = ARIMA(history1, order=orderlist)
                model_fit1 = model1.fit(disp=0)
                output1 = model_fit1.forecast(forecast)
                #print("output1: ")
                #print(output1)
                yhat1 = output1[0]
                yhat1 = yhat1.tolist()
                #print("yhat1: ")
                #print(yhat1)
                predictions1.extend(yhat1)
                #print("predictions")
                #print(predictions)
                #print("test")
                #print(test)
                obs1 = test1[(t1*forecast):(t1*forecast)+forecast:1] #
the actual values for the predicted three weeks. Add this to history
                #print("obs")
                #print(obs)
                history1.extend(obs1)
                #print("history1")
                #print(history1)
     # calculate error of rolling period = forecast length
     #print(test1)
     #print(predictions1)
     error = (sqrt(mean_squared_error(test1, predictions1))) (s.mean(X))
```

Figure 10: Evaluate Models function: ARIMA Forecast.py

Each forecast is outputted as a CSV file. The file contains the actual values, predicted values, and their percent difference for the variable over the test subset of data (most recent 20%

of data). The predicted values of the variable over the forecast duration, as well as 60%, 70%, 80%, and 90% confidence interval values, are supplied as an output as well. See Figure 11 for the 12-week forecast of IP Census Days.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1		Date	Actual	Predictions	Percent Difference	Lower 60% CI	Upper 60% CI	Lower 70% CI	Upper 70% CI	Lower 80% CI	Upper 80% CI	Lower 90% CI	Upper 90% CI
2	0	6/20/2020	768	784.778072	0.021846448								
3	1	6/27/2020	770	779.318655	0.012102149								
29	27	12/26/2020	628	788.608157	0.255745473								
30	28	1/2/2021	689	665.980239	-0.033410393								
31	29	1/9/2021	779	710.799358	-0.087548962								
32	30	1/16/2021	884	745.41743	-0.156767614								
33	31	1/23/2021	916	836.259169	-0.087053308								
34	32			886.872936		845.7217104	928.0241611	836.1963341	937.5495373	824.2112357	949.5346357	806.4475145	967.298357
35	33			886.693833		834.5136506	938.8740148	822.4353743	950.9522911	807.2381448	966.1495206	784.7135613	988.6741041
36	34			861.161354		803.2771565	919.0455524	789.8785575	932.4441513	773.0200606	949.3026483	748.0332285	974.2894804
37	35			873.594567		812.4598009	934.7293337	798.3087848	948.8803498	780.5035752	966.6855594	754.1135725	993.0755621
38	36			873.125865		808.7388514	937.5128789	793.8350295	952.4167008	775.0826184	971.169112	747.2887202	998.9630101
39	37			854.376766		787.1604191	921.5931121	771.6016845	937.1518466	752.0252437	956.7282874	723.0100095	985.7435217
40	38			830.306863		762.0000214	898.613705	746.1888672	914.4248592	726.2948248	934.3189016	696.8088572	963.8048692
41	39			807.688271		737.9744105	877.4021308	721.8375702	893.538971	701.5337405	913.8428008	671.4404062	943.936135
42	40			806.205969		736.0805006	876.3314382	719.8483843	892.5635545	699.4246755	912.9872633	669.1536624	943.2582764
43	41			800.069039		729.8849684	870.2531087	713.6392875	886.4987896	693.1985113	906.9395658	662.9022018	937.2358753
44	42			796.744134		726.5265938	866.9616738	710.2731656	883.2151021	689.8226414	903.6656262	659.511884	933.9763836
45	43			792.389501		722.1296549	862.6493465	705.866434	878.9125674	685.4035885	899.3754129	655.074569	929.7044324

Figure 11: ARIMA CSV 12 Week Forecast Output - Total IP Census Days. Dates 7/4/2020 to 12/19/2020 hidden

With this CSV output, an analyst can easily make a very informative graph, displaying the model's predictions, actual values and forecast with confidence intervals over time. The current forecasting efforts at CCMC use a similar graphing format to the one displayed below. This illustration would be the primary means of sharing the forecast with senior leaders and management. All other graphs for each metric have been placed in Appendix A.

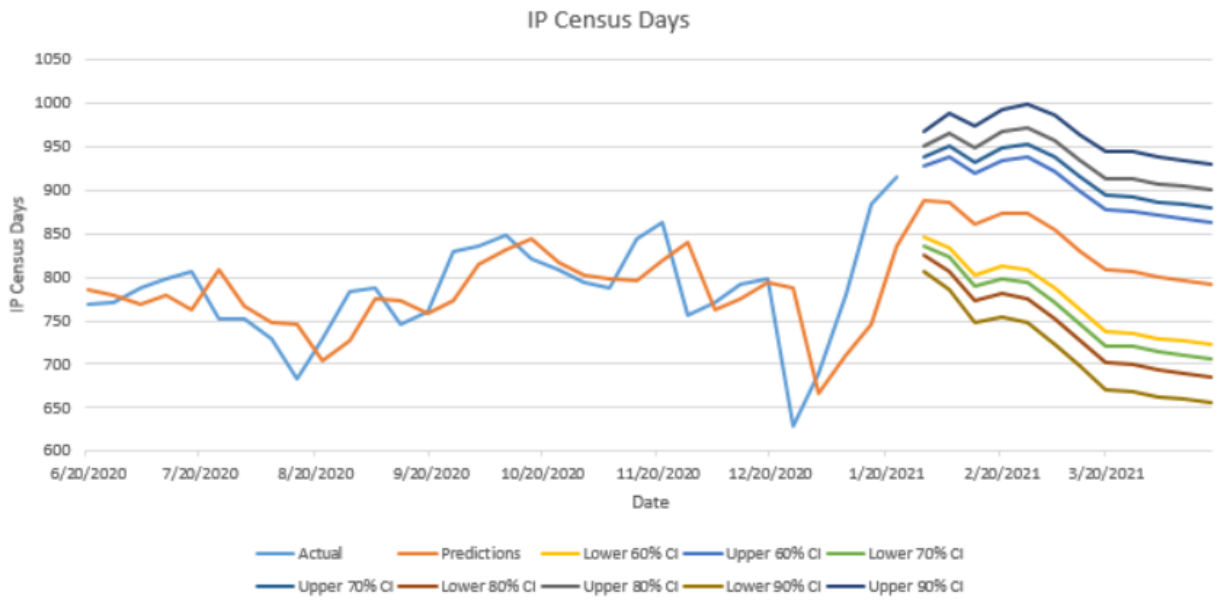


Figure 12: ARIMA Graph Output - IP Census Days

Lastly, the forecast error for the specified forecast lengths is outputted as a .csv file. This allows the analyst to understand the reliability of the forecast as its length increases. See Figure 13:

	A	B	C	D
1		Varible	Normalized Root Mean Squared Error	Forecasted Weeks
2	0	IP_Census.csv	0.08326091	4
3	1	IP_Census.csv	0.053729995	6
4	2	IP_Census.csv	0.044751272	12

Figure 13: Error Output - IP Census Days

From this error output, we see that the NRMSE actually decreases as the forecast length increases. This is not expected, as we would assume that the error would increase as the length increases. This indicates that the IP Census Day's variability is easier to model with ARIMA in 12-week slices than with 4-week slices.

4.1 LIAT Inventory Model

The LIAT test model was created to help understand the forecasted usage of LIAT tests and what timing and patient groups would be applicable for usage. As described in Chapter 3 of this report, the data inputs were collected through available dashboards and communication with

the supply chain team. Historical data on the volume of inpatients, ED direct admits, behavioral health, and urgent OR patients for the past three years were used as the baseline for the model. These historical values were multiplied by modifying values based on expert research to determine forecasted demand. The team then calculated the standard deviation of this forecast and created additional forecasts for high and low demand scenarios based on that. The high demand scenario was representative of 2 standard deviations above the forecasted values, and the low demand scenario was representative of 2 standard deviations below. Figure 14 shows the model results.

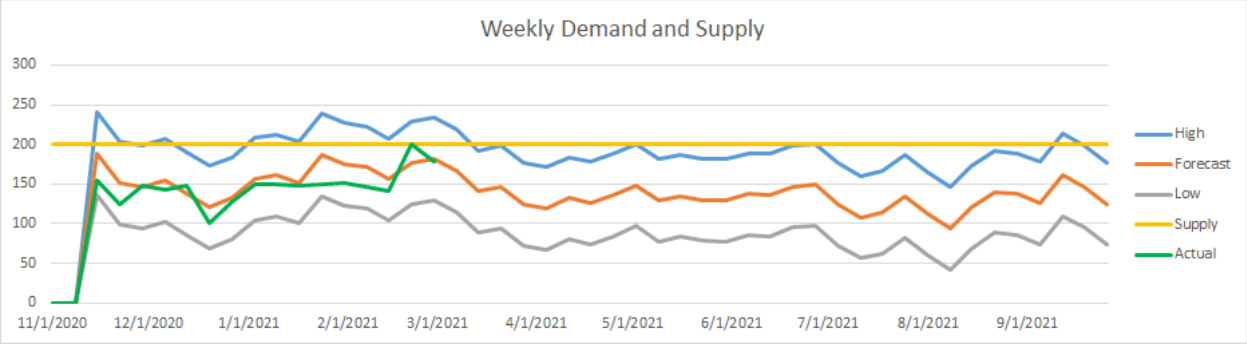


Figure 14: LIAT Model Demand Output

The yellow line Figure 14 is representative of the 200 weekly units of supply that the hospital receives. The orange is the base forecast calculated by the model. The gray line is two standard deviations below that forecast, and the blue line is two standard deviations above that forecast. The green line shows the actual LIAT usage each week. This graph was a promising sign for the team because it demonstrated that LIAT usage was occurring within the bounds of our forecast and commonly centered around the average forecast (orange line). Additionally, it showed that aside from potentially a few weeks in the peak of January and February, the hospital is expected to use less than the 200 per week ration of LIAT tests.

A secondary output of the analysis was graphs of the LIAT inventory level. The LIAT inventory started at 2,560 during the week of 11/15/20, when the first LIAT tests were given. From there, the model creates three graphs, one for each demand scenario, that show the LIAT inventory over time. These graphs can be seen in Figures 15-17 and are labeled for the demand scenario they represent (high medium, or low).

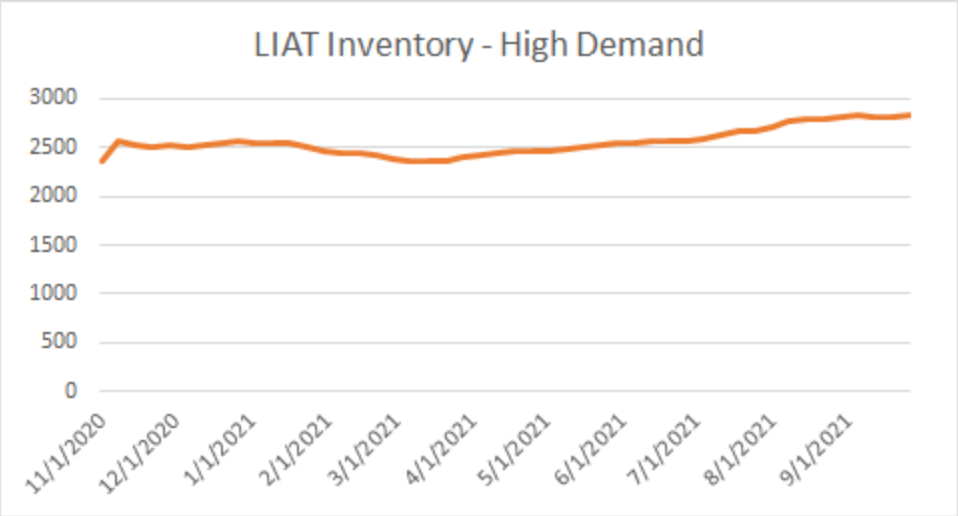


Figure 15: LIAT Model High Demand Inventory Forecast

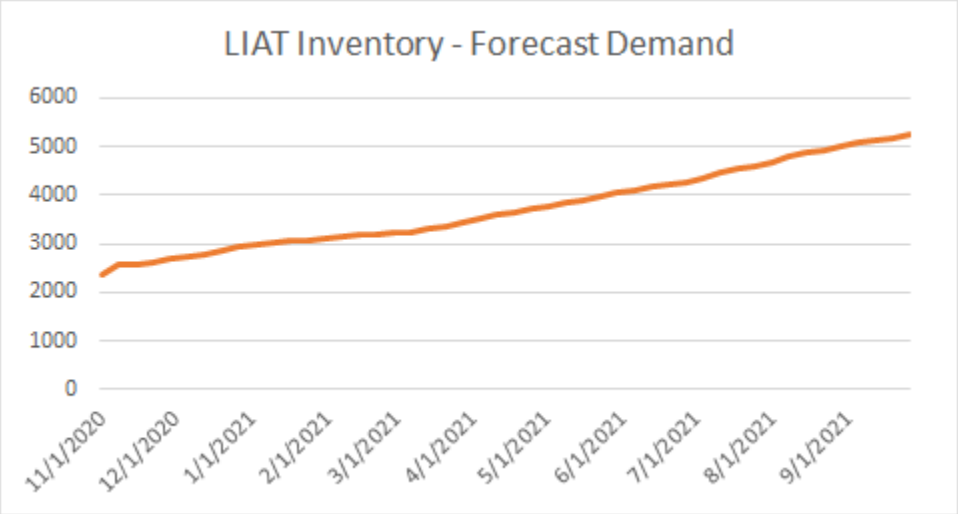


Figure 16: LIAT Model Medium Demand Inventory Forecast

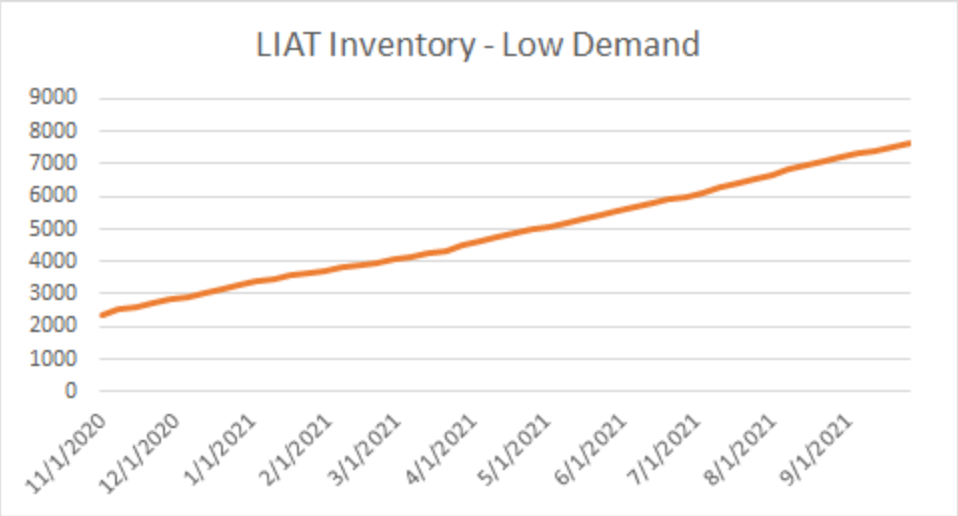


Figure 17: LIAT Model Low Demand Inventory Forecast

In all three graphs, the LIAT inventory eventually is increasing, and even in the high demand scenario, it never dips below 2000 available tests. This information allowed the LIAT team at Connecticut Children's to expand LIAT tests to a larger audience of patient groups. They are currently analyzing which patient groups they can expand to and what those expansion volumes look like. The model can be adjusted dynamically to show the impact that adding certain patient groups has and to verify that the added patient groups will not deplete inventory volumes.

4.3 Behavioral Health Staffing Optimization Model

Deliverables for the behavioral health staffing portion of the project were based around two staffing optimization models, each with its own distinct audience. Both models used data inputs from the RN Acuity live spreadsheet the hospital uses for staffing behavioral health units and excel solver optimization to produce optimal staffing requirements and allocations, as is described in Chapter 3. The first version of the model was geared toward leadership and staffing managers. The main question the model aimed to answer was "What is the optimal number of staffed PCPs, PCAs, and RNs required to meet behavioral health patient sitting needs across all shifts?" The objective function was created and is described as the sum product of staffing costs and the staffing quantities used across the shifts. This objective function is minimized by the excel solver model by changing the decision variables. The decision variables are the staffing

requirements by staffing type, labeled as Max Staffing by Shifts. The following constraints were set on the model:

- Decision variables (staffing numbers) cannot be greater than the maximum achievable staffing for each staffing type and time block
- Decision variables (staffing numbers) must be positive whole numbers
- Bedside coverage (calculated as the total staff divided by the total sitter requirements) must be above the bedside coverage targets for each time block

The solver model is then run and changes the decision variables of staffing requirements to reach the minimum staffing cost that would still meet all constraints set by the model.

The modeled solution for minimum cost for the average weekday scenario for a recent month can be seen in Figure 18. Gray highlighted cells represent outputs and blue highlighted cells represent inputs.

Model Goal - This model attempts to show the minimum staffing cost required to meet 85% compliance to bedside based on various staffing constraints

Objective	
Staffing Cost	$ 10,976.00

Objective Cost Inputs			
PCP	PCP Double	PCA Cost	RN Cost
72	144	76	128

Constraints
Staff are whole numbers
Staff are > 0

Decisions - Staffing Required				
Block	PCPs	PCP Double	PCA Extra	RN Extra
0700-1100	2	1	9	8
1100-1500	3	1	8	8
1500-1900	3	0	9	8
1900-2300	4	0	8	8
2300-0300	5	0	8	4
0300-0700	2	1	8	6
Sum	19	3	50	42

Max Staffing by Shift			
PCPs	PCP Double	PCA Extra	RN Extra
2	1	9	8
3	1	8	8
3	1	9	8
4	1	8	8
5	1	8	8
2	1	8	8
19	6	50	48

Avg Requirements						
Block	ED	MS6	MS7	MS8	PICU	PCPs
0700-1100	6.1	6.7	4.3	2.6	0.2	19.8
1100-1500	6.1	6.7	4.2	2.5	0.2	19.6
1500-1900	6.4	6.8	4.2	2.4	0.3	20.0
1900-2300	6.3	6.9	4.1	2.3	0.3	19.9
2300-0300	6.1	3.8	4.2	2.5	0.2	16.8
0300-0700	5.8	3.7	4.2	2.5	0.2	16.4
Sum	36.8	34.6	25.1	14.7	1.3	112.5

Bedside Coverage		
100.89%	>=	100%
101.93%	>=	100%
100.02%	>=	100%
100.59%	>=	100%
101.35%	>=	100%
103.59%	>=	100%

Figure 18: Average Weekday Costs Recent Month

As can be seen in Figure 18, the optimal solution where all bedside needs are met yields a staffing cost of $10,976 per weekday. The gray matrix shows the optimal staffing requirements based on the hiring constraints. There are also slicers in the model for the month, grid type (weekend or weekday), and the actual dates. These slicers control the pivot table that pulls average sitter requirements and can be dynamically adjusted to rerun the solver problem with different input values. They are shown here in Figure 19.

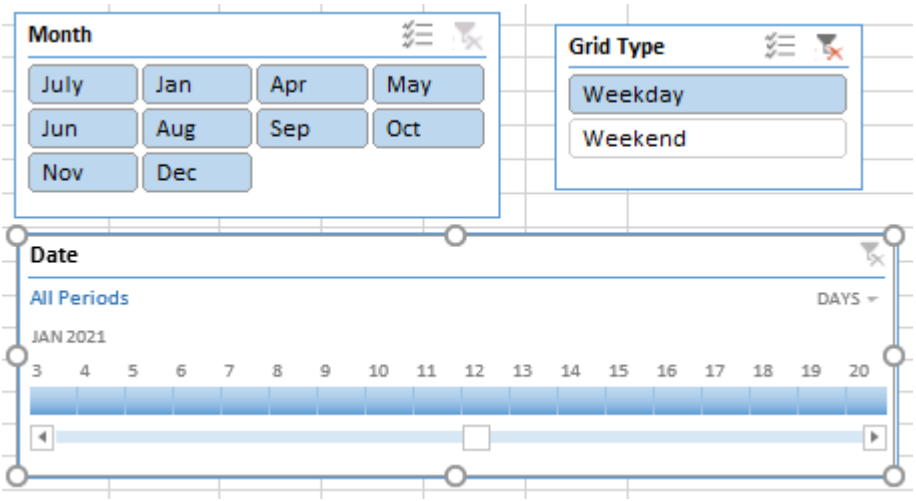


Figure 19: Input Slicer Filters

This solution above focuses on meeting the average sitter requirements on weekdays. Figure 20 below shows the same scenario but for sitter requirements on weekends, resulting in a staffing cost of $11,088 per weekend day.

Model Goal - This model attempts to show the minimum staffing cost required to meet 85% compliance to bedside based on various staffing constraints

Objective	
Staffing Cost	$ 11,088.00

Objective Cost Inputs			
PCP	PCP Double	PCA Cost	RN Cost
72	144	76	128

Constraints
Staff are whole numbers
Staff are > 0

Decisions - Staffing Required				
Block	PCPs	PCP Double	PCA Extra	RN Extra
0700-1100	2	1	9	8
1100-1500	3	1	8	8
1500-1900	3	0	9	8
1900-2300	4	0	8	7
2300-0300	5	0	8	5
0300-0700	2	0	8	8
Sum	19	2	50	44

Max Staffing by Shift			
PCPs	PCP Double	PCA Extra	RN Extra
2	1	9	8
3	1	8	8
3	1	9	8
4	1	8	8
5	1	8	8
2	1	8	8
19	6	50	48

Avg Requirements						
Block	ED	MS6	MS7	MS8	PICU	PCPs
0700-1100	6.6	6.7	4.0	2.1	0.5	19.9
1100-1500	6.6	6.6	4.0	2.1	0.5	19.8
1500-1900	7.0	5.8	4.2	2.6	0.2	19.8
1900-2300	6.6	5.4	4.3	2.3	0.1	18.7
2300-0300	6.8	4.0	4.0	2.3	0.3	17.4
0300-0700	6.6	4.1	4.0	2.3	0.3	17.3
Sum	40.1	32.6	24.5	13.8	1.9	112.9

Bedside Coverage		
100.45%	>=	100%
100.90%	>=	100%
101.19%	>=	100%
101.45%	>=	100%
103.67%	>=	100%
103.98%	>=	100%

Figure 20: Average Weekend Costs Recent Month

These scenario model outputs showing average weekday and weekend requirements will allow CCMC to determine where they should focus on hiring needs. With that being said, there are hundreds of combinations of inputs that could be altered to produce many different results and outputs. That fluidity is a key aspect of the model as a whole and allows for comparing different scenarios. One such comparison that the team performed for CCMC was to look at the cost impact of being able to fully hire patient care partners so that no extra PCAs, RNs, or double-time PCPs need to be pulled away from their roles to sit with patients. The results of that scenario for weekdays and weekends can be seen in Figure 21 and Figure 22.

Model Goal - This model attempts to show the minimum staffing cost required to meet 85% compliance to bedside based on various staffing constraints

Objective	
Staffing Cost	$ 8,208.00

Objective Cost Inputs			
PCP	PCP Double	PCA Cost	RN Cost
72	144	76	128

Constraints
Staff are whole numbers
Staff are > 0

Decisions - Staffing Required				
Block	PCPs	PCP Double	PCA Extra	RN Extra
0700-1100	20	0	0	0
1100-1500	20	0	0	0
1500-1900	20	0	0	0
1900-2300	20	0	0	0
2300-0300	17	0	0	0
0300-0700	17	0	0	0
Sum	114	0	0	0

Max Staffing by Shift			
PCPs	PCP Double	PCA Extra	RN Extra
30	1	9	8
30	1	8	8
30	1	9	8
30	1	8	8
30	1	8	8
30	1	8	8
180	6	50	48

Avg Requirements						
Block	ED	MS6	MS7	MS8	PICU	PCPs
0700-1100	6.1	6.7	4.3	2.6	0.2	19.8
1100-1500	6.1	6.7	4.2	2.5	0.2	19.6
1500-1900	6.4	6.8	4.2	2.4	0.3	20.0
1900-2300	6.3	6.9	4.1	2.3	0.3	19.9
2300-0300	6.1	3.8	4.2	2.5	0.2	16.8
0300-0700	5.8	3.7	4.2	2.5	0.2	16.4
Sum	36.8	34.6	25.1	14.7	1.3	112.5

Bedside Coverage		
100.89%	>=	100%
101.93%	>=	100%
100.02%	>=	100%
100.59%	>=	100%
101.35%	>=	100%
103.59%	>=	100%

Figure 21: Average Weekday Costs Recent Month with all PCPs

Model Goal - This model attempts to show the minimum staffing cost required to meet 85% compliance to bedside based on various staffing constraints

Objective	
Staffing Cost	$ 8,280.00

Objective Cost Inputs			
PCP	PCP Double	PCA Cost	RN Cost
72	144	76	128

Constraints
Staff are whole numbers
Staff are > 0

Decisions - Staffing Required				
Block	PCPs	PCP Double	PCA Extra	RN Extra
0700-1100	20	0	0	0
1100-1500	20	0	0	0
1500-1900	20	0	0	0
1900-2300	19	0	0	0
2300-0300	18	0	0	0
0300-0700	18	0	0	0
Sum	115	0	0	0

Max Staffing by Shift			
PCPs	PCP Double	PCA Extra	RN Extra
30	1	9	8
30	1	8	8
30	1	9	8
30	1	8	8
30	1	8	8
30	1	8	8
180	6	50	48

Avg Requirements						
Block	ED	MS6	MS7	MS8	PICU	PCPs
0700-1100	6.6	6.7	4.0	2.1	0.5	19.9
1100-1500	6.6	6.6	4.0	2.1	0.5	19.8
1500-1900	7.0	5.8	4.2	2.6	0.2	19.8
1900-2300	6.6	5.4	4.3	2.3	0.1	18.7
2300-0300	6.8	4.0	4.0	2.3	0.3	17.4
0300-0700	6.6	4.1	4.0	2.3	0.3	17.3
Sum	40.1	32.6	24.5	13.8	1.9	112.9

Bedside Coverage		
100.45%	>=	100%
100.90%	>=	100%
101.19%	>=	100%
101.45%	>=	100%
103.67%	>=	100%
103.98%	>=	100%

Figure 22: Average Weekend Costs Recent Month with all PCPs

For weekdays, the total cost per day is reduced from $10,976 to $8,208, which is a total savings of $2,768 per day, translating to roughly $719,680 per year. For weekend requirements, the cost is reduced from $11,088 to $8,280, which is a total of $2,808 per day, translating to $292,032 per year. Therefore, the total cost savings for fully hiring PCPs and avoiding the use of PCAs or RNs would accumulate to $1,011,712 per year. This cost savings assume that PCAs, RNs, and double-time PCPs are treated as an additional cost when in actuality, they are performing these duties on top of their usual roles. For that reason, a change like this would not save CCMC that much money directly but rather free up that many dollars' worth of resources to return to performing the responsibilities they were hired for. Additionally, these hiring volumes are unrealistic given the high turnover rate for PCPs. In that sense, the cost association provided is an example of what potential savings could be in an ideal state, but several other factors and analyses would be required to perform an in-depth and accurate cost analysis.

This is an example of just one type of scenario analysis that can be performed with this model. Different scenarios that could be run include (but are not limited to) changing bedside coverage targets, changing sitter requirements based on forecasted growth, or applying weights to the different time blocks based on typical volumes. All of these could be dynamically changed

and rerun with the solver model without having to make any development changes to the model itself.

The second version of the model focuses on the question, “What is the minimum shortage of patient sitters that can be optimally achieved given the current staffing model?” The results from this model are meant to be used on a daily basis. As described in the methodology, the model takes in the sitter requirements and staffing constraints and assigns staff to departments in order to achieve the minimum staffing shortage while applying volume-based weighting. The use case for this model is therefore focused on operations rather than leadership and hiring managers. The objective function in this model is a minimization of the weighted shortages by department and time block. It is calculated as the sum of squares for all shortages by department. The solver model optimizes that value by adjusting staffing allocation decision variables labeled “Decisions - Staffing Required.” The model constraints are as follows.

- The sum of staffing by time block must not exceed the user constraint of available staffing for that time block
- All shortages are greater than or equal to 0 (no negative shortages)
- All staffing allocations must be whole numbers greater than or equal to 0

Output based on sample data from a recently given day can be seen in Figure 23.

Model Goal - This model attempts to show the optimal staffing allocation to minimize the weighted shortage of PCPs by department

Objective	
Sumsq Weighted Short	103.00
Actual Short	47

Constraints	
Short > = 0	0

Decisions - Suggested Staffing Allocation

Block	ED	MS6	MS7	MS8	PICU	PCPs		Sum	Available Staffing by Shift: PCPs	PCA Extra	RN Extra
0700-1100	2	5	0	0	0	7	=	7	5	2	0
1100-1500	3	5	0	0	0	8	=	8	6	2	0
1500-1900	1	5	1	0	0	7	=	7	5	2	0
1900-2300	2	4	1	1	0	8	=	8	6	2	0
2300-0300	4	2	2	2	0	10	=	10	8	2	0
0300-0700	4	2	1	1	0	8	=	8	6	2	0
Sum	16	23	5	4	0	48		48	36	12	0

PCPs Short

Block	ED	MS6	MS7	MS8	PICU	Total
0700-1100	3	3	3	2	0	11
1100-1500	2	3	3	2	0	10
1500-1900	2	1	2	2	0	7
1900-2300	2	2	2	1	0	7
2300-0300	1	2	1	1	0	5
0300-0700	1	2	2	2	0	7
Avg	11	13	13	10	0	47

PCP Requirements

Block	ED	MS6	MS7	MS8	PICU	PCPs
0700-1100	5	8	3	2	0	18
1100-1500	5	8	3	2	0	18
1500-1900	3	6	3	2	0	14
1900-2300	4	6	3	2	0	15
2300-0300	5	4	3	3	0	15
0300-0700	5	4	3	3	0	15
Sum	27	36	18	14	0	95

Figure 23: Staff Shortage Optimization Output for a Given Day

As can be seen in Figure 23, the minimum sum of squares for shortages with the given constraints on staffing and sitter requirements was 103, with the actual shortage being 47 PCPs. This value was achieved by the staffing allocations matrix shown in gray. Additionally, the shortages by department and by time block are shown in the #PCPs Short matrix. They are conditionally formatted where green represents no shortages, yellow represents a shortage of 1, and red represents any shortage greater than 1. This kind of analysis could be replicated for multiple days or over the data set as an average as well. It allows for CCMC to see where their resources should be allocated in order to match patient volumes and what the shortages for each area are. The data from each day could also be manually saved and analyzed separately to examine typical shortages over an extended period of time.

5.0 Recommendations

The recommendations section discusses the recommendations that the team made to CCMC in regard to ARIMA forecasting, LIAT testing, and behavioral health staffing. In short, the ARIMA forecast recommendations relate to a proposed implementation plan and process flow for the ARIMA demand forecasts. The LIAT testing recommendations involve expanding the eligible patient groups for testing and looking to use this model as a template for future inventory and demand challenges that are similar in nature. The behavioral health staffing recommendations involve further expansion of the prototyped optimization models in order to dig deeper into optimizing behavioral health staffing to meet patient needs at a minimum cost.

5.1 ARIMA Demand Forecasting

At the time of writing, the ARIMA model has been developed, tested, and approved by the Enterprise Analytics Team as an accurate forecasting method. However, it currently consists of two python scripts and a handful of SQL queries, which still require some manual effort to run (2 hours a forecasting cycle). We would recommend that an end-to-end forecasting system is implemented, requiring limited human interaction to run the ARIMA model. In this section, we present the end-to-end system that we have prototyped, as well as include some implementation steps. We believe the recommendations that follow in this section are well worth Connecticut Children's Enterprise Analytics Team's time and energy.

In the first phase of the model's use, we recommend only using it on the ten variables that we developed the model for. This will provide a small subset of data to test the implementation's functionality. In order for the model to provide value to CC, the system should run the model automatically on a specified interval. The system will have a primary end-user, denoted as an analyst, as well as a model maintainer. The analyst will provide the inputs as needed, and the model maintainer will implement the system and fix any issues that arise. For a full description of the responsibilities of both the analyst and the model maintainer, see Appendix A. Figure 24 illustrates the business process and information flow for the implemented system. Note the ARIMA Model is denoted as 'Model.'

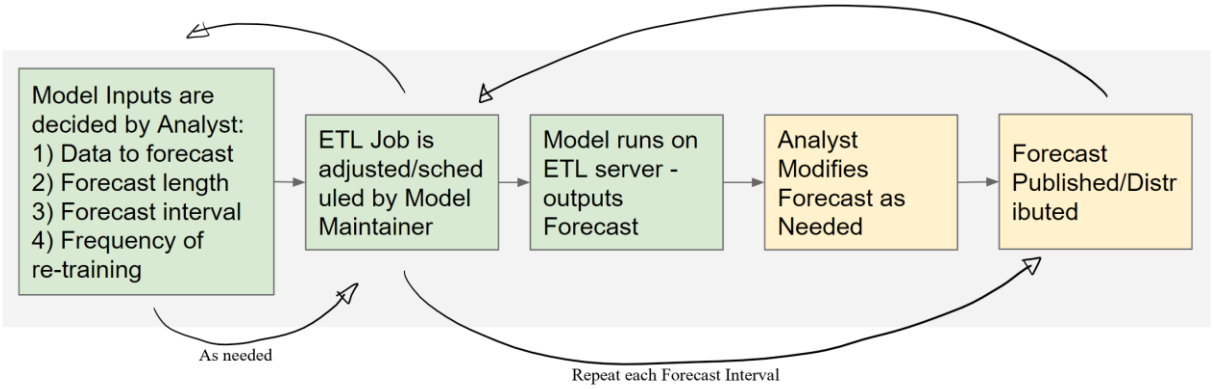


Figure 24: Integrating the Model into CCMC Processes

Data to forecast: the data of interest. Also commonly referred to as a variable.
Forecast Length: number of units of time that a value is predicted.
Forecast Interval: the frequency that a new forecast is generated.
Frequency of re-training: how often the model's re-parameterized

For each data .csv (data value we want to forecast) there will be a separate ETL script and SQL Query/View. The analyst will decide the forecast length, forecast interval, and frequency of re-training for that data .csv. The ETL job will first run the *Grid Search ARIMA Parameterization.py* with the data .csv and forecast length as inputs. Once this script has run, the output is a .csv with the best P, D, and Q values. Then, the *ARIMA Forecast.py* script should be run, with the P, D, and Q values, data .csv, and range of forecast length as inputs. This will output .csvs for each forecast length containing the forecast.

Once the model has output a forecast, we believe that the analyst should then modify the forecast as they see fit. After all, the ARIMA model does have its limitations. The model only knows what has happened in the past and uses that information to predict the future. Connecticut Children's Enterprise Analytics department currently gets market research information from external sources, as well as information from healthcare experts, about potential health care trends in the future. After extensive conversations with an analyst and manager in the Enterprise Analytics Department, the team believes that the health care trends should be included in the forecast. Since they would be impossible to integrate into the model, the analyst would be responsible for adjusting the forecast using the external healthcare trend information.

However, modifying the forecast does make it more difficult to track the error of the forecast over time. The current system tracks the error of the model each time the model is run.

Tracking the model's error over time should be done by the system. Similarly, tracking the forecast's error after the analyst has adjusted it should also be done by the system to give the analyst an understanding of the impact of their forecast adjustments.

As shown in section 4.1, the output of the model can be graphed to share with others. Visualizing the forecast is a very important step, as raw numbers do not convey much for regular audiences. If possible, we believe that the visualization could also be automated and published in a Business Intelligence report. This would allow the model to be viewed more frequently by more Connecticut Children's personnel.

There is quite some debate as to whether or not the *Grid Search ARIMA Parameterization.py* should be run each time a new forecast is produced, or if the P, D, and Q values should be saved and reused until the error of the forecasts degrades a substantial amount. Since the *Grid Search ARIMA Parameterization.py* uses the most recent 20% of data to identify the best P, D, and Q values, it would be sensible to re-parameterize each time. However, the *Grid Search ARIMA Parameterization.py* script uses a lot of memory and performance, so it may not be feasible to run it often. Hence, this debate will continue, but the error should be monitored, and re-parameterization should occur as often as possible.

5.2 LIAT Inventory

Based on the results of the LIAT inventory model, the team has several recommendations. The first recommendation was made with the LIAT inventory management team. Since there were plenty of available LIAT tests and inventory did not appear to be at risk in either the short or long term, the team was able to recommend expansion to additional patient groups. Expanding to other patient groups will allow the hospital to process some patients more rapidly and with fewer restrictions due to the quick LIAT results. This allows for more available space and fewer staffing requirements in order to process patients. Currently, the LIAT management team is analyzing which patient groups are best to expand to base on need and volume, and they can use the current model to aid their analysis.

Additionally, the modeling process using this methodology provides a good foundation for future modeling efforts. In the future, situations like LIAT testing can now be analyzed using

similar forecasting techniques and models. This can be especially applicable for different types of PPE or other inventory challenges within the hospital. The LIAT model could even be used as a template and would likely require only slight modifications to adapt to a different scenario. The key recommendation this team can make is to continue modeling in a dynamic and automated way similar to the LIAT model. The flexibility of this type of modeling provides is critical for scenario analysis in the ever-changing environment of healthcare.

5.3 Behavioral Health Staffing Optimization

There are several key recommendations for the behavioral health staffing model that the team developed as well. The first, and broadest, is that analysis efforts like this need to be expanded further for behavioral health. The patient volumes continue to grow. Staffing and space requirements need to match that growth where possible. Where that isn't possible, further analysis and continuous improvement efforts will be critical for the shortages. Specifically, the results from the first version of the model provide a clear recommendation that hiring sitters to meet demand has the potential to be a critical cost savings effort for the hospital, with potential yearly savings of about $1,011,712. It also ensures PCAs, and RNs are able to perform their normal responsibilities without needing to fill in as PCPs. They also should carefully examine their current staffing compared to what the optimized staffing should be in order to potentially shift resources where needed. This could be especially beneficial when looking at weekday vs. weekend inputs. Further examinations and scenarios related to other factors such as PCP employee turnover rate could also be very beneficial to optimizing behavioral health costs.

On the operations side, the team recommends that continuous improvement experts in behavioral health begin to use the model to support operational analysis. They can use the model on a daily basis to determine the optimal allocation of PCP resources based on patient volume. This would allow them to prove the resources are being split as fairly as possible between each department. They can also use this model to analyze the shortages over a longer period of time to examine where the most common gaps are and identify potential solutions.

Finally, a complete and thorough implementation of these models would be an extremely beneficial future project. They are set up to be used for running scenarios and prototyping to aid

in decision-making. However, if they could be fully integrated into the leadership and operations decision-making processes, they could make significant impacts on many decisions. Having a resource comfortable in the use of excel solver to make more complex changes to the model would be an extremely valuable asset to both the leadership and floor operations teams. These efforts would be a great opportunity for a future capstone project for other students from Worcester Polytechnic Institute or another academic organization.

6.0 Conclusion

The project team developed analytical tools in three major areas for CCMC: Demand Forecasting, LIAT Testing Inventory, and Behavioral Health Staffing. The ARIMA Demand Forecasting provided CC with a fully automated forecasting system. Alongside data enterprise experts at CCMC, the team developed and documented a forecasting methodology for ten key metrics. The model uses the ARIMA methodology to provide weekly forecast volumes for a given data value. The outputs are .csv files which can be adjusted by data analysts and experts based on market trends and other information before being put into graphical displays. Different forecast ranges based on confidence intervals are included as well to capture the impact of variability. Overall, the ARIMA methodology and modeling that the team has been able to provide will serve as a baseline for data analysts for reporting variable forecasts, providing significant time savings from the manual process they were using prior. The effort itself is expandable to other variables if the need arises.

For LIAT testing, the team was able to provide a forecasting model that could predict patient volumes and compare them to the supply and inventory of LIAT Tests. The model uses a fixed, rationed supply of tests and patient volumes from the four patient groups currently receiving the tests. The model results showed that the patient volumes consistently stayed below the supply of 200 LIAT tests per week. Given the preliminary stockpile of inventory and demand for tests being below weekly supply, the team recommended that LIAT tests extend to additional patient groups. The CCMC LIAT team is currently using the model to analyze which groups that tests should and can be expanded to without depleting inventory. LIAT testing will eventually become much less of a critical factor in the hospital, so the model itself will not require long-term automation or implementation. It can be used as a template to expand to other resouces such as PPE.

The behavioral health staffing optimization model provided critical insights into the current state of staffing Patient Care Partners (PCPs) in the behavioral health units of CCMC. PCPs sit with behavioral health patients to ensure safety and proper care. The solver optimization models aimed to answer two questions. The first was “What is the optimal number of staffed PCPs, PCAs, and RNs required to meet behavioral health patient sitting needs across all shifts?”

The team was able to use the model to provide recommendations for the optimal staffing of PCPs and show the cost impacts of using other roles such as PCAs and RNs to cover typical PCP duties. The second question the models aimed to answer was “What is the minimum shortage of patient sitters that can be optimally achieved given the current staffing model?” The team built a model that answers this question by taking inputs for current staffing and showing the optimal allocation of those resources to various departments. This model could be implemented and automatically tied to the live PCP requirements spreadsheet and used as a daily planning tool for the behavioral health floors. That implementation process is outside the project scope but could be a valuable future project.

Overall, the analytical and modeling efforts in all three of these areas highlighted the value of industrial engineering and analytical tools in making decisions within the hospital. In all three areas, the models produced outputs that impacted choices made within the hospital. Additionally, the project was able to show several of the opportunities that exist within CCMC for this type of work. This project team merely scratched the surface of the current analytics technology. Full-time industrial engineers or comparable full-time equivalents could significantly impact these projects and expand their impact. Being able to make decisions based on automated analytical tools like the ones in this project will continue to positively impact the hospital and its quality of care it can provide to its patients.

7.0 Appendices

7.1 Appendix A - ARIMA Python Scripts

Grid Search ARIMA Parameterization.py

```
### Developer: Jack Tanny (jetanny@wpi.edu) and Jihan Nabahani
(jnabahani@wpi.edu).
### Last Modified: 2/9/2020
### Purpose: Develop the best p, d, and q parameters of an ARIMA
time-series forecasting model
### Required Inputs: A csv file with one column holding the time a
data point was collected, and
# another column with the corresponding data point

### Last Modified: 2/19/2020 - jetanny

#////////////////////////////////////////////////////////////////
////////////////////////////////

### Import Libraries with Versions
import pandas as pd
# Version:  1.0.5  pandas is a popular Python-based data analysis
toolkit. It presents a diverse range of utilities, ranging from
parsing multiple file formats to converting an entire data table into
a NumPy matrix array
# link:https://docs.anaconda.com/anaconda/navigator/tutorials/pandas/
import statistics as s
# To access Python statistics functions such as mean & standard
deviations
from pandas import read_csv
# imports data as a tibble (a modern reimagining of the data)
from datetime import datetime
# It basically sets datetime to be a reference to the class, then
immediately setting it to be a reference to the module
from pandas import DataFrame
# Two-dimensional labeled data structures with columns of potentially
different types generally 3 main components: data, the index, and the
```

```
columns
from pandas import DatetimeIndex
# One of pandas date offset strings or corresponding objects, can be
passed in order to set the frequency of the index as the inferred
frequency upon creation
from statsmodels.tsa.arima_model import ARIMA
# Version 0.11.1 statsmodels is a basic interface for ARIMA-type
models, including those with exogenous regressors and those with
seasonal components.
from matplotlib import pyplot as plt
from sklearn.metrics import mean_squared_error
from pandas.plotting import autocorrelation_plot
# version 3.2.2 matplot is a collection of command style functions
that make matplotlib work like MATLAB. creates a figure, creates a
plotting area in a figure, plots some lines in a plotting area, etc
from math import sqrt
# will import the square root function into the current namespac
import numpy as np
# version 1.18.5  NumPy is a general-purpose array-processing
package. It provides a high-performance multidimensional array
object, and tools for working with these arrays. It is shortened to
np to make code easier
import csv
# CSV files are used to store a large number of variables or data.
They are incredibly simplified spreadsheets such as Excel
import warnings as warnings
# to keep warnings from appearing during the running of the below
functions.

## Additional Libraries needed to evaluate ARIMA(p,d,q) fit:
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

# To get date
from datetime import date as d

## Sources
# adapted from https://machinelearningmastery.com/arima-for-time-
series-forecasting-with-python/
```

```
# adapted from https://towardsdatascience.com/arima-forecasting-in-python-
90d36c2246d3#:~:text=ARIMA%20stands%20for%20Autoregressive%20Integrated,errors%20%2F%20order%20of%20MA%20terms

#?????????????????????????????????????????????????????????????????????
??????????????????????????

# define evaluate_arima_model. This function evaluates an ARIMA model on a dataset X for a given arima_order (p,d,q)
def evaluate_arima_model(X, arima_order):
     X = series.values
     # making the split proportions for the training and testing data
     size = int(len(series)*0.8)  # making the split proportions. Using oldest 80% for training and the rest (newest 20%) for testing
     train, test = X[0:size], X[size:len(X)]  # traing and testing split
     #print(train)
     #print(test)
     #print(X)
     test = test.tolist()
     history = [x for x in train] # create history list. This will store the historical values (rolling) during the for loop.
     #print(type(history))
     predictions = list()
     #print(len(test)//4) # will always round down

     ############################################################
     forecast = 6 #set the forecast value to the lenght of forecast that you will make using ARIMA Forecast
     ############################################################

     for t in range(len(test)//forecast): # for loop that loops through the test dataset, making a new ARIMA forecast every 'forecast' weeks.
          # i.e. if forecast = 6, this will loop through the test set every 6 weeks, make a forecast, and record the error
          for o in range(forecast-1):
```

```
                if len(test)/forecast != len(test)//forecast:
                      test.pop() # removes the last occurance of test
                      #print(test)
          if t==0:
                model = ARIMA(train, order=arima_order)
                model_fit = model.fit(disp=0)
                output = model_fit.forecast(forecast)
                #print("output: ")
                #print(output)
                yhat = output[0] # get the guesses for the next three
weeks
                yhat = yhat.tolist() # make guesses into a list
                #print("yhat: ")
                #print(yhat)
                predictions.extend(yhat) # add three guesses into
predictions
                #print("predictions")
                #print(predictions)
                #print("test")
                #print(test)
                obs = test[t:t+forecast:1] # get the next three
actual values from test
                #print("obs")
                #print(obs)
                history.extend(obs) # add to history with obs
                #print("history")
                #print(history)
          else:
                #print("predictions :")
                #print(predictions)
                model = ARIMA(history, order=arima_order)
                model_fit = model.fit(disp=0)
                output = model_fit.forecast(forecast)
                #print("output: ")
                #print(output)
                yhat = output[0]
                yhat = yhat.tolist()
                #print("yhat: ")
                #print(yhat)
```

```
                predictions.extend(yhat)
                #print("predictions")
                #print(predictions)
                #print("test")
                #print(test)
                obs = test[(t*forecast):(t*forecast)+forecast:1] #
the actual values for the predicted three weeks. Add this to history
                #print("obs")
                #print(obs)
                history.extend(obs)
                #print("history")
                #print(history)
     error = (sqrt(mean_squared_error(test,
predictions)))/(s.mean(X))
     return error

#????????????????????????????????????????????????????????????????????
??????????????????????????

## define evaluate_models, which evaluates all defined combinations
of p, d and q values of an ARIMA model on the passed dataset
def evaluate_models(dataset, p_values, d_values, q_values):
     dataset = dataset.astype('float32')
     best_score, best_cfg = float("inf"), None
     for p in p_values:
          for d in d_values:
               for q in q_values:
                    order = (p,d,q)
                    try:
                         mse = evaluate_arima_model(dataset, order)
                         if mse < best_score:
                              best_score, best_cfg = mse, order
                         print('ARIMA%s MSE=%.3f' % (order,mse))
                    except:
                         continue
     print('Best ARIMA%s MSE=%.3f' % (best_cfg, best_score))
     return(best_cfg)

def parser(x):
```

```
      return datetime.strptime(x,'%m/%d/%Y')

# pass the CSV file that you want the best p, d, and q paramters for
csvs = 'G:\\WPIMQP2020\\ARIMAModels\\ARIMANEW 1 25
2021\\IP_Census.csv'

errorlist = [] #list to store error values
variablenamelist = [] #list to store variable names
orderlist = [] #list stores the best p, d, q values

variablesplit = csvs.split('\\')
variablename = variablesplit[-1] # get the file name from the list of
split parts of the input file path. This should be the only file
variablenamelist.append(variablename)
series = pd.read_csv(csvs, parse_dates=[0], index_col=0,
squeeze=True, date_parser=parser)

# look at data
print(series.head())
date = series.index
print(date)
datelist = []
datelist.append(date.strftime('%m/%d/%Y').tolist())
print(datelist[0])
datelist1 = datelist[0]
series.plot()
plt.show() # if this plt has a clear trend, we know it will need some
differencing of at least one

## Statistical tests for verifying p, d, and q manually

# ADF test for stationary
# if this is greater than 0.05, it needs differencing
#print("p-value:", adfuller(series.values))

# finding the differencing term, d
#fig = plt.figure(figsize=(10, 10))
#ax1 = fig.add_subplot(311)
#fig = plot_acf(series, ax=ax1, title="Autocorrelation")
```

```
#ax2 = fig.add_subplot(312)
#fig = plot_acf(series.diff().dropna(), ax=ax2,
                #title="1st Order Differencing")
#ax3 = fig.add_subplot(313)
#fig = plot_acf(series.diff().diff().dropna(), ax=ax3,
           #title="2nd Order Differencing")
#plt.show()  # which ever order has only 1 very significant lag above
the CI, then thats the order of differencing

# finding the p term to make data stationary using partial
autocorrelation
#plot_pacf(series.diff().dropna(), lags=40)
#plt.show()

# finding the q term to make data stationary using autocorrelation
#plot_acf(series.diff().dropna())
#plt.show()  # if this plot shows a statistically significant
autocorrelation for x lags, the AR can start at x


# define ranges for ARIMA grid search

p_values = [0, 1, 2, 4, 6, 8, 10]
d_values = range(0, 3)
q_values = range(0, 3)
warnings.filterwarnings("ignore")
bestorder = evaluate_models(series.values, p_values, d_values,
q_values)
print(bestorder)
orderlist.append(bestorder)

# # fit model - see how well the best_cfg has worked
# model = ARIMA(series, order=best_cfg)
# model_fit = model.fit(disp=0)
# model_fit.plot_predict(1,len(series.index)+5)
# print(model_fit.summary())
# # plot residual errors
# residuals = DataFrame(model_fit.resid)
# residuals.plot()
```

```
# plt.show()
# residuals.plot(kind='kde')
# plt.show()
# print(residuals.describe())

# print to CSV
path = 'H:\\'
print(variablenamelist)
print(orderlist)
pandaerrorlist = pd.DataFrame({'Varible': variablenamelist, 'P, D,
Q': orderlist})
print(pandaerrorlist)
elfilename = 'bestparametersanderror' + str(d.today()) + variablename
print(path+elfilename)
pandaerrorlist.to_csv(path+elfilename)
```

ARIMA Forecast.py

```
### Developer: Jack Tanny (jetanny@wpi.edu) and Jihan Nabahani
(jnabahani@wpi.edu).
### Last Modified: 2/9/2020
### Purpose: Develop the best p, d, and q parameters of an ARIMA
time-series forecasting model
### Required Inputs: A csv file with one column holding the time a
data point was collected, and
# another column with the corresponding data point

### Last Modified: 2/19/2020 - jetanny

#////////////////////////////////////////////////////////////////
////////////////////////////////

### Import Libraries with Versions
import pandas as pd
# Version:  1.0.5  pandas is a popular Python-based data analysis
toolkit. It presents a diverse range of utilities, ranging from
parsing multiple file formats to converting an entire data table into
a NumPy matrix array
```

```
# link:https://docs.anaconda.com/anaconda/navigator/tutorials/pandas/
import statistics as s
# To access Python statistics functions such as mean & standard
deviations
from pandas import read_csv
# imports data as a tibble (a modern reimagining of the data)
from datetime import datetime
# It basically sets datetime to be a reference to the class, then
immediately setting it to be a reference to the module
from pandas import DataFrame
# Two-dimensional labeled data structures with columns of potentially
different types generally 3 main components: data, the index, and the
columns
from pandas import DatetimeIndex
# One of pandas date offset strings or corresponding objects, can be
passed in order to set the frequency of the index as the inferred
frequency upon creation
from statsmodels.tsa.arima_model import ARIMA
# Version 0.11.1 statsmodels is a basic interface for ARIMA-type
models, including those with exogenous regressors and those with
seasonal components.
from matplotlib import pyplot as plt
from sklearn.metrics import mean_squared_error
from pandas.plotting import autocorrelation_plot
# version 3.2.2 matplot is a collection of command style functions
that make matplotlib work like MATLAB. creates a figure, creates a
plotting area in a figure, plots some lines in a plotting area, etc
from math import sqrt
# will import the square root function into the current namespac
import numpy as np
# version 1.18.5  NumPy is a general-purpose array-processing
package. It provides a high-performance multidimensional array
object, and tools for working with these arrays. It is shortened to
np to make code easier
import csv
# CSV files are used to store a large number of variables or data.
They are incredibly simplified spreadsheets such as Excel
import warnings as warnings
# to keep warnings from appearing during the running of the below
```

```
functions.

## Additional Libraries needed to evaluate ARIMA(p,d,q) fit:
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

# To get date
from datetime import date as d

## Sources
# adapted from https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/
# adapted from https://towardsdatascience.com/arima-forecasting-in-python-90d36c2246d3#:~:text=ARIMA%20stands%20for%20Autoregressive%20Integrated,errors%20%2F%20order%20of%20MA%20terms

#?????????????????????????????????????????????????????????????????????
??????????????????????????

# define evaluate_arima_model. This function evaluates an ARIMA model on a dataset X for a given arima_order (p,d,q)
def evaluate_arima_model(X, arima_order):
     X = series.values
     # making the split proportions for the training and testing data
     size = int(len(series)*0.8)  # making the split proportions. Using oldest 80% for training and the rest (newest 20%) for testing
     train, test = X[0:size], X[size:len(X)]  # traing and testing split
     #print(train)
     #print(test)
     #print(X)
     test = test.tolist()
     history = [x for x in train] # create history list. This will store the historical values (rolling) during the for loop.
     #print(type(history))
     predictions = list()
     #print(len(test)//4) # will always round down
```

```
     ############################################################
     forecast = 6 #set the forecast value to the lenght of forecast
that you will make using ARIMA Forecast
     ############################################################

     for t in range(len(test)//forecast): # for loop that loops
through the test dataset, making a new ARIMA forecast every
'forecast' weeks.
          # i.e. if forecast = 6, this will loop through the test
set every 6 weeks, make a forecast, and record the error
          for o in range(forecast-1):
               if len(test)/forecast != len(test)//forecast:
                    test.pop() # removes the last occurance of test
                    #print(test)
          if t==0:
               model = ARIMA(train, order=arima_order)
               model_fit = model.fit(disp=0)
               output = model_fit.forecast(forecast)
               #print("output: ")
               #print(output)
               yhat = output[0] # get the guesses for the next three
weeks
               yhat = yhat.tolist() # make guesses into a list
               #print("yhat: ")
               #print(yhat)
               predictions.extend(yhat) # add three guesses into
predictions
               #print("predictions")
               #print(predictions)
               #print("test")
               #print(test)
               obs = test[t:t+forecast:1] # get the next three
actual values from test
               #print("obs")
               #print(obs)
               history.extend(obs) # add to history with obs
               #print("history")
               #print(history)
```

```
            else:
                  #print("predictions :")
                  #print(predictions)
                  model = ARIMA(history, order=arima_order)
                  model_fit = model.fit(disp=0)
                  output = model_fit.forecast(forecast)
                  #print("output: ")
                  #print(output)
                  yhat = output[0]
                  yhat = yhat.tolist()
                  #print("yhat: ")
                  #print(yhat)
                  predictions.extend(yhat)
                  #print("predictions")
                  #print(predictions)
                  #print("test")
                  #print(test)
                  obs = test[(t*forecast):(t*forecast)+forecast:1] #
the actual values for the predicted three weeks. Add this to history
                  #print("obs")
                  #print(obs)
                  history.extend(obs)
                  #print("history")
                  #print(history)
      error = (sqrt(mean_squared_error(test,
predictions)))/(s.mean(X))
      return error

#????????????????????????????????????????????????????????????????????
??????????????????????????????

## define evaluate_models, which evaluates all defined combinations
of p, d and q values of an ARIMA model on the passed dataset
def evaluate_models(dataset, p_values, d_values, q_values):
      dataset = dataset.astype('float32')
      best_score, best_cfg = float("inf"), None
      for p in p_values:
            for d in d_values:
                  for q in q_values:
```

```
                        order = (p,d,q)
                        try:
                              mse = evaluate_arima_model(dataset, order)
                              if mse < best_score:
                                    best_score, best_cfg = mse, order
                              print('ARIMA%s MSE=%.3f' % (order,mse))
                        except:
                              continue
      print('Best ARIMA%s MSE=%.3f' % (best_cfg, best_score))
      return(best_cfg)

def parser(x):
      return datetime.strptime(x,'%m/%d/%Y')

# pass the CSV file that you want the best p, d, and q paramters for
csvs = 'G:\\WPIMQP2020\\ARIMAModels\\ARIMANEW 1 25
2021\\IP_Census.csv'

errorlist = [] #list to store error values
variablenamelist = [] #list to store variable names
orderlist = [] #list stores the best p, d, q values

variablesplit = csvs.split('\\')
variablename = variablesplit[-1] # get the file name from the list of
split parts of the input file path. This should be the only file
variablenamelist.append(variablename)
series = pd.read_csv(csvs, parse_dates=[0], index_col=0,
squeeze=True, date_parser=parser)

# look at data
print(series.head())
date = series.index
print(date)
datelist = []
datelist.append(date.strftime('%m/%d/%Y').tolist())
print(datelist[0])
datelist1 = datelist[0]
series.plot()
plt.show() # if this plt has a clear trend, we know it will need some
```

```
differencing of at least one

## Statistical tests for verifying p, d, and q manually

# ADF test for stationary
# if this is greater than 0.05, it needs differencing
#print("p-value:", adfuller(series.values))

# finding the differencing term, d
#fig = plt.figure(figsize=(10, 10))
#ax1 = fig.add_subplot(311)
#fig = plot_acf(series, ax=ax1, title="Autocorrelation")
#ax2 = fig.add_subplot(312)
#fig = plot_acf(series.diff().dropna(), ax=ax2,
                #title="1st Order Differencing")
#ax3 = fig.add_subplot(313)
#fig = plot_acf(series.diff().diff().dropna(), ax=ax3,
           #title="2nd Order Differencing")
#plt.show()  # which ever order has only 1 very significant lag above
the CI, then thats the order of differencing

# finding the p term to make data stationary using partial
autocorrelation
#plot_pacf(series.diff().dropna(), lags=40)
#plt.show()

# finding the q term to make data stationary using autocorrelation
#plot_acf(series.diff().dropna())
#plt.show()  # if this plot shows a statistically significant
autocorrelation for x lags, the AR can start at x


# define ranges for ARIMA grid search

p_values = [0, 1, 2, 4, 6, 8, 10]
d_values = range(0, 3)
q_values = range(0, 3)
warnings.filterwarnings("ignore")
bestorder = evaluate_models(series.values, p_values, d_values,
```

```
q_values)
print(bestorder)
orderlist.append(bestorder)

# # fit model - see how well the best_cfg has worked
# model = ARIMA(series, order=best_cfg)
# model_fit = model.fit(disp=0)
# model_fit.plot_predict(1,len(series.index)+5)
# print(model_fit.summary())
# # plot residual errors
# residuals = DataFrame(model_fit.resid)
# residuals.plot()
# plt.show()
# residuals.plot(kind='kde')
# plt.show()
# print(residuals.describe())

# print to CSV
path = 'H:\\'
print(variablenamelist)
print(orderlist)
pandaerrorlist = pd.DataFrame({'Varible': variablenamelist, 'P, D,
Q': orderlist})
print(pandaerrorlist)
elfilename = 'bestparametersanderror' + str(d.today()) + variablename
print(path+elfilename)
pandaerrorlist.to_csv(path+elfilename)
```

7.2 Appendix B - ARIMA Job Descriptions

The ARIMA Forecasting Model developed by the 2020/2021 WPI MQP team has two primary actors: 1) Model Maintainer 2) Analyst. It is completely possible that one person will satisfy the responsibilities of both actors. However, that is not necessary.

Model Maintainer: 4 hours/forecast cycle

Required Skills:

- Working knowledge with Python 3 programming language.
- Intermediate knowledge with SQL Queries and ETL Scripting
- Working knowledge of the CC data warehouse and databases
- Basics of Machine Learning and Statistics

Recommended Skills (can be easily learned/taught)

- ARIMA in Python
    - https://www.statsmodels.org/devel/generated/statsmodels.tsa.arima_model.ARIMA.html
    - https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/
    - https://www.datacamp.com/courses/arima-models-in-python
- Strong quantitative analytical skills

Commitment:

We estimate that it will take 10 hours for the model maintainer to ramp up and learn the model. The system, as-is of March 2021, will require at most 2 hours of model maintenance per forecast cycle. In other words, if the forecasts are generated monthly, the model maintainer will have 2 hours of work per month. The 2 hours would most likely be spent fixing any bugs that come up, adjusting the ETL Job/SQL Queries if the data cataloging changes, and investigating data sources as needed. These actions would be conducted at request of the analyst.

If the model were to be expanded to other forecast variables, the model maintainer's commitment would increase linearly. If the model were to be modified to fit a new use case, the model maintainer would be the one to make the necessary modifications.

Analyst:

Required Skills:

- Strong quantitative analytics
- Critical thinking and stakeholder management
- Ability to break down and communicate complex ideas and influence senior leaders.
- Basic Excel data visualization

Recommended Skills (Can be taught):

- Time-Series Machine Learning basics
- Experience in the data enterprising team

Commitment:

Similar to the model developer, the analyst will need to ramp up to the role. This will take anywhere from 5-20 hours, depending on their familiarity with forecasting, statistics, and the organization.

Once they are ramped up, it will take 1 hour every forecasting cycle to prepare visualizations for the 10 variables the as-is system forecasts. It will take another 1-3 hours to adjust the forecast output to account for information that is not captured in time-series forecasting (i.e. market research, information from senior leaders, health care trends). An additional 1 hour/forecast cycle should be reserved for investigating unexpected increases/decreases in error. The analyst could then spend anywhere from 0-10 hours a forecasting cycle communicating the forecasts to senior leaders. This will be at the discretion of Renee Blanchard, Director of Enterprise Analytics. It is important to note that many of these activities could already be the responsibility of a CC employee that is close to the data enterprising team.

7.3 Appendix C - ARIMA Forecast Graphs

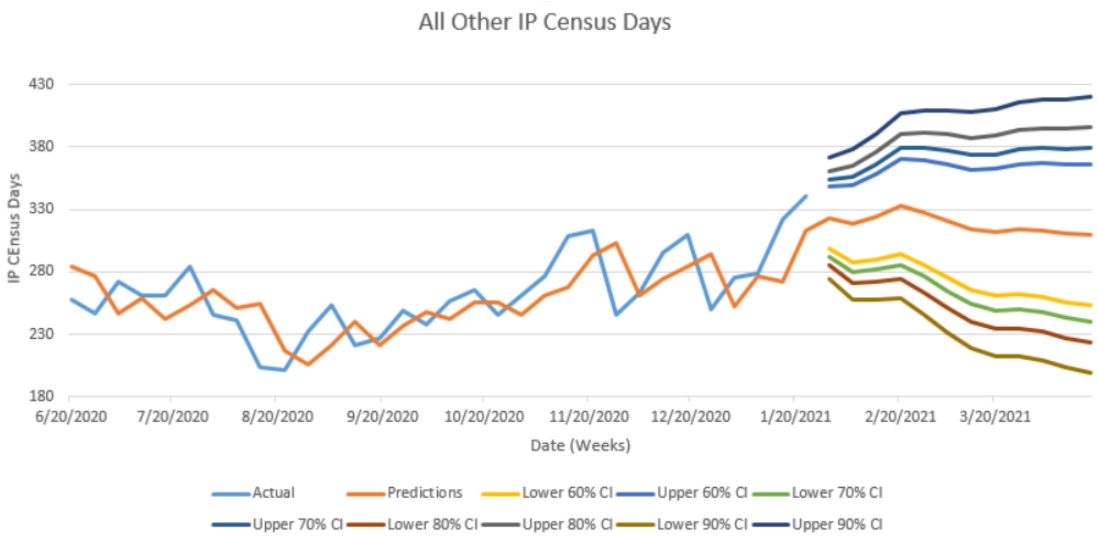


Figure A-1: ARIMA Forecast All Other IP Census Days

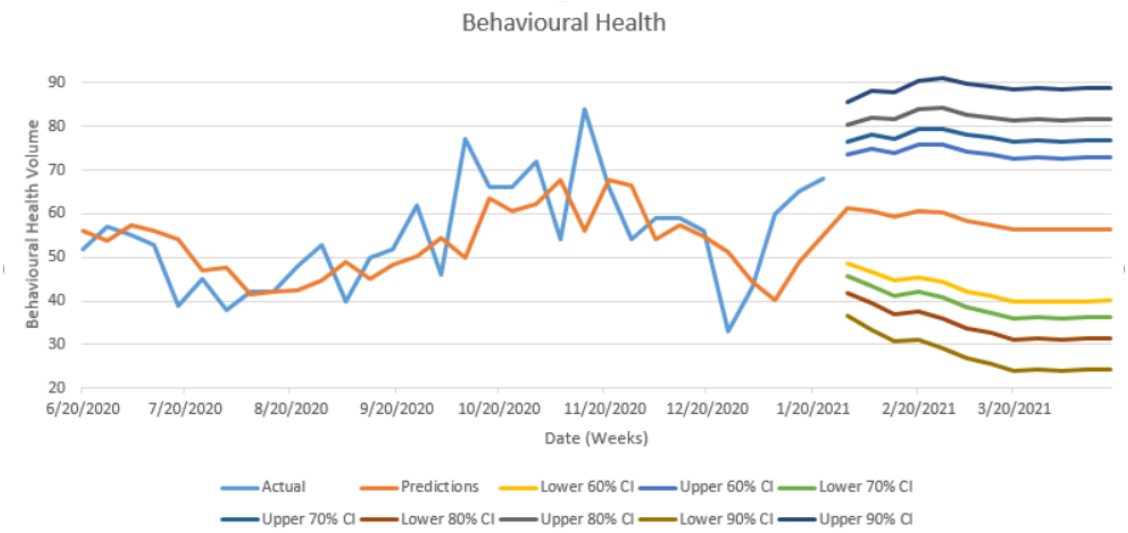


Figure A-2: ARIMA Forecast Behavioral Health

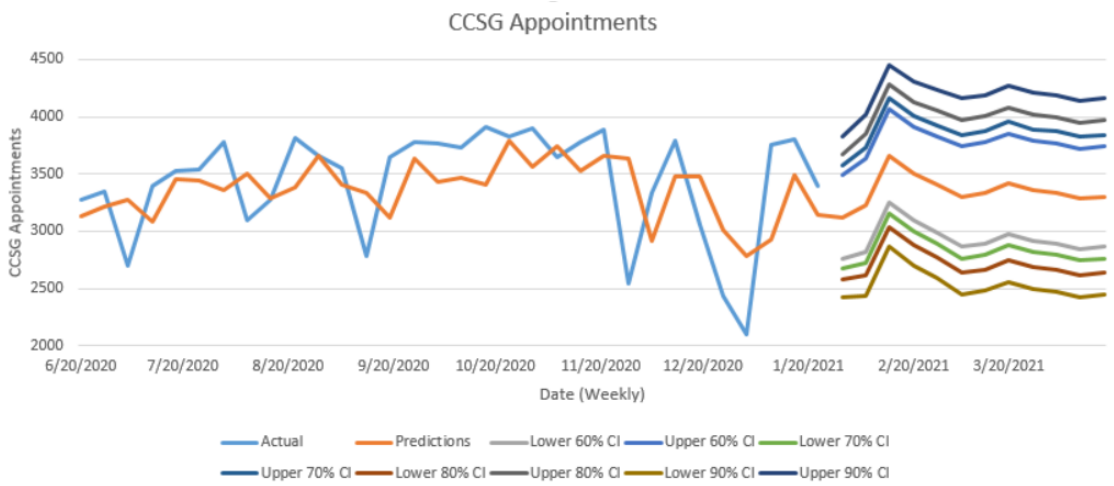


Figure A-3: ARIMA Forecast CCSG Appointments

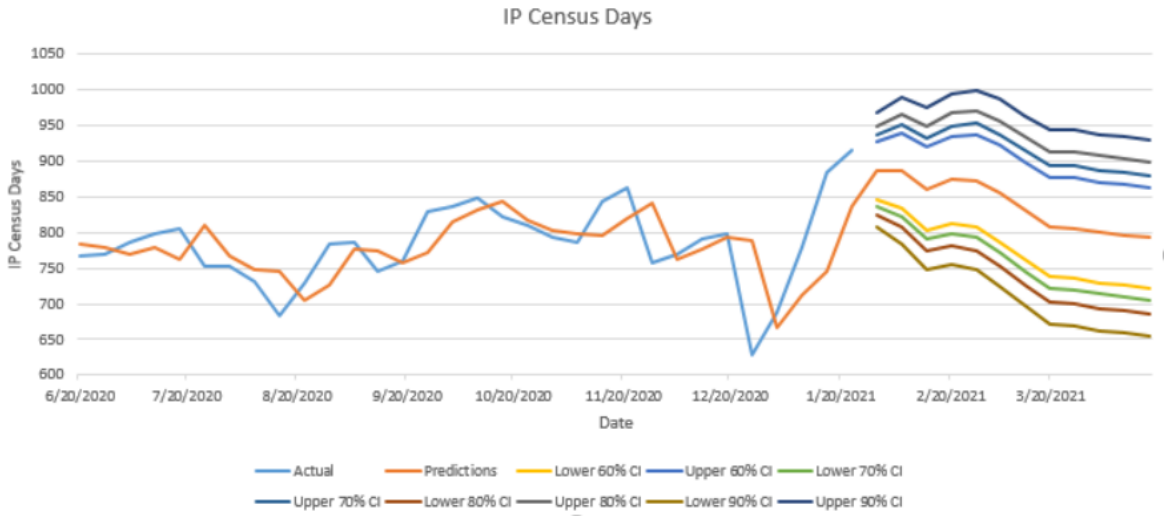


Figure A-4: ARIMA Forecast IP Census Days

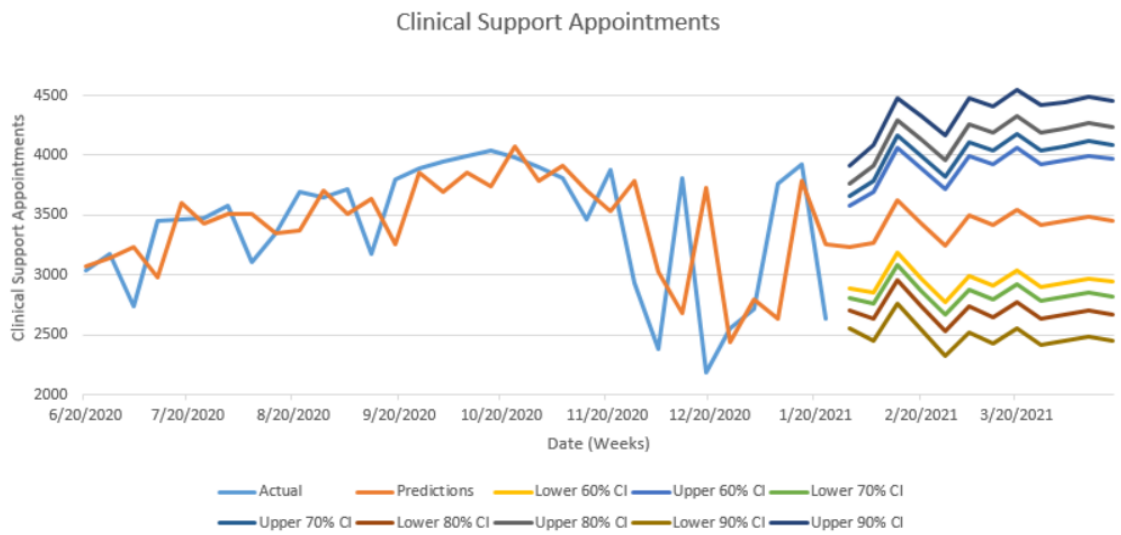


Figure A-5: ARIMA Forecast Clinical Support Appointments

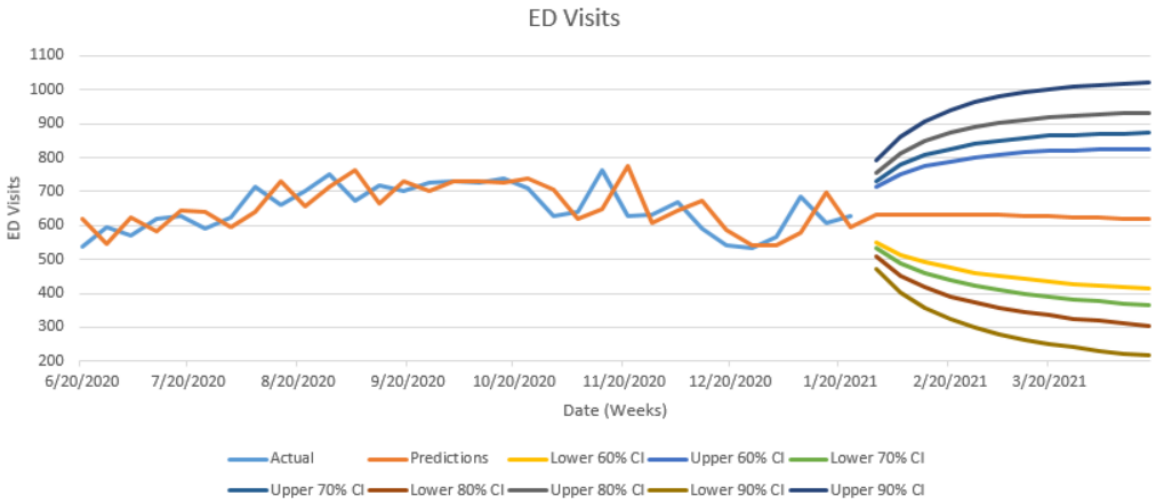


Figure A-6: ARIMA Forecast ED Visits

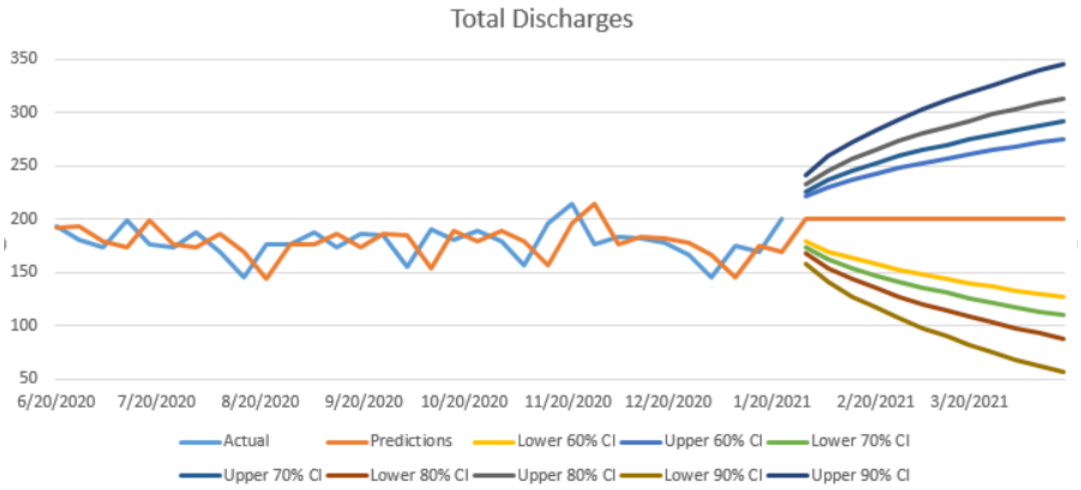


Figure A-7: ARIMA Forecast Total Discharges

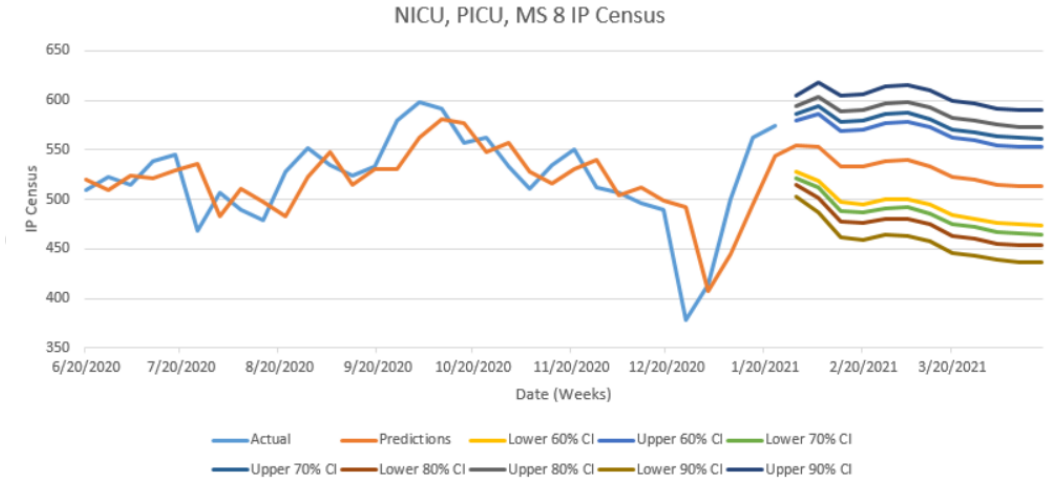


Figure A-8: ARIMA Forecast NICU, PICU and MS8 IP Census Days

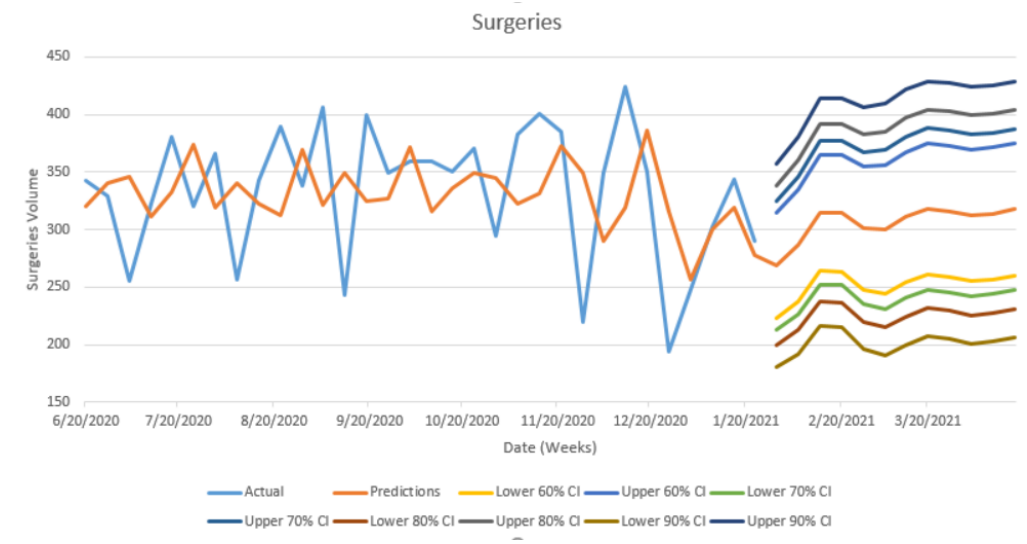


Figure A-9: ARIMA Forecast Surgeries

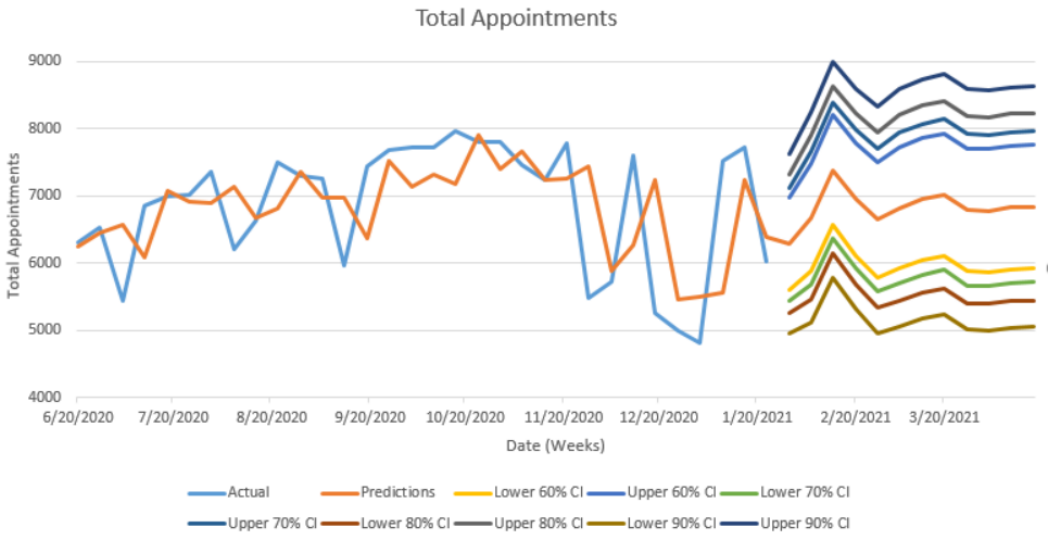


Figure A-10: ARIMA Forecast Total Appointments

7.4 Appendix D - LIAT Testing Model

	FY2020 Historical							FY 2021 Predictions		
	Inpatients	Behavioral Health	ED OP Admissions	Urgent OR	Total FY 2020	% Change		2020-2021 Forecast	Upper	Lower
12/9/2019 - 12/15/2019	76	89	20	8	=SUM(B13:E13)	0.8	=I12+7	=F13*G13	=J13+2*$P$4	=J13-2*$P$4
12/16/2019 - 12/22/2019	86	59	5	22	=SUM(B14:E14)	0.8	=I13+7	=F14*G14	=J14+2*$P$4	=J14-2*$P$4
12/23/2019 - 12/29/2019	88	51	7	5	=SUM(B15:E15)	0.8	=I14+7	=F15*G15	=J15+2*$P$4	=J15-2*$P$4
12/30/2019 - 1/5/2020	81	55	8	21	=SUM(B16:E16)	0.8	=I15+7	=F16*G16	=J16+2*$P$4	=J16-2*$P$4
1/6/2020 - 1/12/2020	84	94	16	1	=SUM(B17:E17)	0.8	=I16+7	=F17*G17	=J17+2*$P$4	=J17-2*$P$4
1/13/2020 - 1/19/2020	100	72	17	12	=SUM(B18:E18)	0.8	=I17+7	=F18*G18	=J18+2*$P$4	=J18-2*$P$4
1/20/2020 - 1/26/2020	98	77	6	9	=SUM(B19:E19)	0.8	=I18+7	=F19*G19	=J19+2*$P$4	=J19-2*$P$4
1/27/2020 - 2/2/2020	89	107	20	17	=SUM(B20:E20)	0.8	=I19+7	=F20*G20	=J20+2*$P$4	=J20-2*$P$4
2/3/2020 - 2/9/2020	86	105	23	4	=SUM(B21:E21)	0.8	=I20+7	=F21*G21	=J21+2*$P$4	=J21-2*$P$4
2/10/2020 - 2/16/2020	78	84	22	6	=SUM(B22:E22)	0.9	=I21+7	=F22*G22	=J22+2*$P$4	=J22-2*$P$4
2/17/2020 - 2/23/2020	73	82	13	5	=SUM(B23:E23)	0.9	=I22+7	=F23*G23	=J23+2*$P$4	=J23-2*$P$4
2/24/2020 - 3/1/2020	72	82	16	7	=SUM(B24:E24)	1	=I23+7	=F24*G24	=J24+2*$P$4	=J24-2*$P$4
3/2/2020 - 3/8/2020	63	91	7	21	=SUM(B25:E25)	1	=I24+7	=F25*G25	=J25+2*$P$4	=J25-2*$P$4
3/9/2020 - 3/15/2020	58	95	10	4	=SUM(B26:E26)	1	=I25+7	=F26*G26	=J26+2*$P$4	=J26-2*$P$4
3/16/2020 - 3/22/2020	39	37	2		=SUM(B27:E27)	1.8	=I26+7	=F27*G27	=J27+2*$P$4	=J27-2*$P$4
3/23/2020 - 3/29/2020	28	39	9	5	=SUM(B28:E28)	1.8	=I27+7	=F28*G28	=J28+2*$P$4	=J28-2*$P$4
3/30/2020 - 4/5/2020	31	27	9	2	=SUM(B29:E29)	1.8	=I28+7	=F29*G29	=J29+2*$P$4	=J29-2*$P$4
4/6/2020 - 4/12/2020	25	29	12		=SUM(B30:E30)	1.8	=I29+7	=F30*G30	=J30+2*$P$4	=J30-2*$P$4
4/13/2020 - 4/19/2020	35	36	14	3	=SUM(B31:E31)	1.5	=I30+7	=F31*G31	=J31+2*$P$4	=J31-2*$P$4
4/20/2020 - 4/26/2020	34	36	14		=SUM(B32:E32)	1.5	=I31+7	=F32*G32	=J32+2*$P$4	=J32-2*$P$4
4/27/2020 - 5/3/2020	37	37	16	1	=SUM(B33:E33)	1.5	=I32+7	=F33*G33	=J33+2*$P$4	=J33-2*$P$4
5/4/2020 - 5/10/2020	28	42	27	2	=SUM(B34:E34)	1.5	=I33+7	=F34*G34	=J34+2*$P$4	=J34-2*$P$4
5/11/2020 - 5/17/2020	53	47	25	4	=SUM(B35:E35)	1	=I34+7	=F35*G35	=J35+2*$P$4	=J35-2*$P$4

Figure D-1: Demand Sheet Formulas

	Demand			Supply	Actual	Inventory			Starting Inventory	2160
	High	Forecast	Low			High	Medium	Low		
44136	0	0	0	200	0	=$O$2+$F$4-C4	=$O$2+$F$4-D4	=$O$2+$F$4-E4		
44143	0	0	0	200	0	=I4+$F5-C5	=J4+$F5-D5	=K4+$F5-E5		
44150	=Demand!K10	=Demand!J10	=Demand!L10	200	154	=I5+$F6-C6	=J5+$F6-D6	=K5+$F6-E6		
44157	=Demand!K11	=Demand!J11	=Demand!L11	200	125	=I6+$F7-C7	=J6+$F7-D7	=K6+$F7-E7		
44164	=Demand!K12	=Demand!J12	=Demand!L12	200	148	=I7+$F8-C8	=J7+$F8-D8	=K7+$F8-E8		
44171	=Demand!K13	=Demand!J13	=Demand!L13	200	142	=I8+$F9-C9	=J8+$F9-D9	=K8+$F9-E9		
44178	=Demand!K14	=Demand!J14	=Demand!L14	200	148	=I9+$F10-C10	=J9+$F10-D10	=K9+$F10-E10		
44185	=Demand!K15	=Demand!J15	=Demand!L15	200	100	=I10+$F11-C11	=J10+$F11-D11	=K10+$F11-E11		
44192	=Demand!K16	=Demand!J16	=Demand!L16	200	127	=I11+$F12-C12	=J11+$F12-D12	=K11+$F12-E12		
44199	=Demand!K17	=Demand!J17	=Demand!L17	200	149	=I12+$F13-C13	=J12+$F13-D13	=K12+$F13-E13		
44206	=Demand!K18	=Demand!J18	=Demand!L18	200	150	=I13+$F14-C14	=J13+$F14-D14	=K13+$F14-E14		
44213	=Demand!K19	=Demand!J19	=Demand!L19	200	148	=I14+$F15-C15	=J14+$F15-D15	=K14+$F15-E15		
44220	=Demand!K20	=Demand!J20	=Demand!L20	200	149	=I15+$F16-C16	=J15+$F16-D16	=K15+$F16-E16		
44227	=Demand!K21	=Demand!J21	=Demand!L21	200	151	=I16+$F17-C17	=J16+$F17-D17	=K16+$F17-E17		
44234	=Demand!K22	=Demand!J22	=Demand!L22	200	147	=I17+$F18-C18	=J17+$F18-D18	=K17+$F18-E18		
44241	=Demand!K23	=Demand!J23	=Demand!L23	200	141	=I18+$F19-C19	=J18+$F19-D19	=K18+$F19-E19		
44248	=Demand!K24	=Demand!J24	=Demand!L24	200	200	=I19+$F20-C20	=J19+$F20-D20	=K19+$F20-E20		
44255	=Demand!K25	=Demand!J25	=Demand!L25	200	179	=I20+$F21-C21	=J20+$F21-D21	=K20+$F21-E21		
44262	=Demand!K26	=Demand!J26	=Demand!L26	200		=I21+$F22-C22	=J21+$F22-D22	=K21+$F22-E22		
44269	=Demand!K27	=Demand!J27	=Demand!L27	200		=I22+$F23-C23	=J22+$F23-D23	=K22+$F23-E23		
44276	=Demand!K28	=Demand!J28	=Demand!L28	200		=I23+$F24-C24	=J23+$F24-D24	=K23+$F24-E24		

FIgure D-2: Model Sheet Calculation Formulas

7.5 Appendix E - Behavioral Health Optimization Models

Model Goal – This model attempts to show the minimum staffing cost required to meet 85% compliance to bedside based on

Objective	
Staffing Cost	=SUMPRODUCT(C17:F17,E7:H7)

Objective Cost Inputs			
PCP	PCP Double	PCA Cost	RN Cost
=18*4	=E7*2	=19*4	=32*4

Decisions – Staffing Required

Block	PCPs	PCP Double	PCA Extra	RN Extra
0700-1100	20	0	0	0
1100-1500	20	0	0	0
1500-1900	20	0	0	0
1900-2300	20	0	0	0
2300-0300	17	0	0	0
0300-0700	17	0	0	0
Sum	=SUM(C11:C16)	=SUM(D11:D16)	=SUM(E11:E16)	=SUM(F11:F16)

Avg Requirements

Block	ED	MS6	MS7	MS8	PICU	PCPs
0700-1100	=GETPIVOTDATA("Average of 1:1 Sits Required",'Sitter Req'!$B$2,"Dept",C$21,"Timeblock",$B22)	=GETPIVOTDA1	=GETPIVOTDATA	=GETPIVOTDATAI	=GETPIVOTDAT,	=SUM(C22:G22)
1100-1500	=GETPIVOTDATA("Average of 1:1 Sits Required",'Sitter Req'!$B$2,"Dept",C$21,"Timeblock",$B23)	=GETPIVOTDA1	=GETPIVOTDATA	=GETPIVOTDATAI	=GETPIVOTDAT,	=SUM(C23:G23)
1500-1900	=GETPIVOTDATA("Average of 1:1 Sits Required",'Sitter Req'!$B$2,"Dept",C$21,"Timeblock",$B24)	=GETPIVOTDA1	=GETPIVOTDATA	=GETPIVOTDATAI	=GETPIVOTDAT,	=SUM(C24:G24)
1900-2300	=GETPIVOTDATA("Average of 1:1 Sits Required",'Sitter Req'!$B$2,"Dept",C$21,"Timeblock",$B25)	=GETPIVOTDA1	=GETPIVOTDATA	=GETPIVOTDATAI	=GETPIVOTDAT,	=SUM(C25:G25)
2300-0300	=GETPIVOTDATA("Average of 1:1 Sits Required",'Sitter Req'!$B$2,"Dept",C$21,"Timeblock",$B26)	=GETPIVOTDA1	=GETPIVOTDATA	=GETPIVOTDATAI	=GETPIVOTDAT,	=SUM(C26:G26)
0300-0700	=GETPIVOTDATA("Average of 1:1 Sits Required",'Sitter Req'!$B$2,"Dept",C$21,"Timeblock",$B27)	=GETPIVOTDA1	=GETPIVOTDATA	=GETPIVOTDATAI	=GETPIVOTDAT,	=SUM(C27:G27)
Sum	=SUM(C22:C27)	=SUM(D22:D27	=SUM(E22:E27)	=SUM(F22:F27)	=SUM(G22:G27)	=SUM(H22:H27)

Figure E-1: Model 1 Formulas Displayed – Decisions

various staffing constraints

Constraints
Staff are whole numbers
Staff are > 0

Max Staffing by Shift			
PCPs	PCP Double	PCA Extra	RN Extra
30	1	9	8
30	1	8	8
30	1	9	8
30	1	8	8
30	1	8	8
30	1	8	8
=SUM(J11:J16)	=SUM(K11:K16)	=SUM(L11:L16)	=SUM(M11:M16)

Bedside Coverage		
=SUM(C11:F11)/H22	>=	1
=SUM(C12:F12)/H23	>=	1
=SUM(C13:F13)/H24	>=	1
=SUM(C14:F14)/H25	>=	1
=SUM(C15:F15)/H26	>=	1
=SUM(C16:F16)/H27	>=	1

Figure E-2: Model 1 Formulas Displayed - Constraints

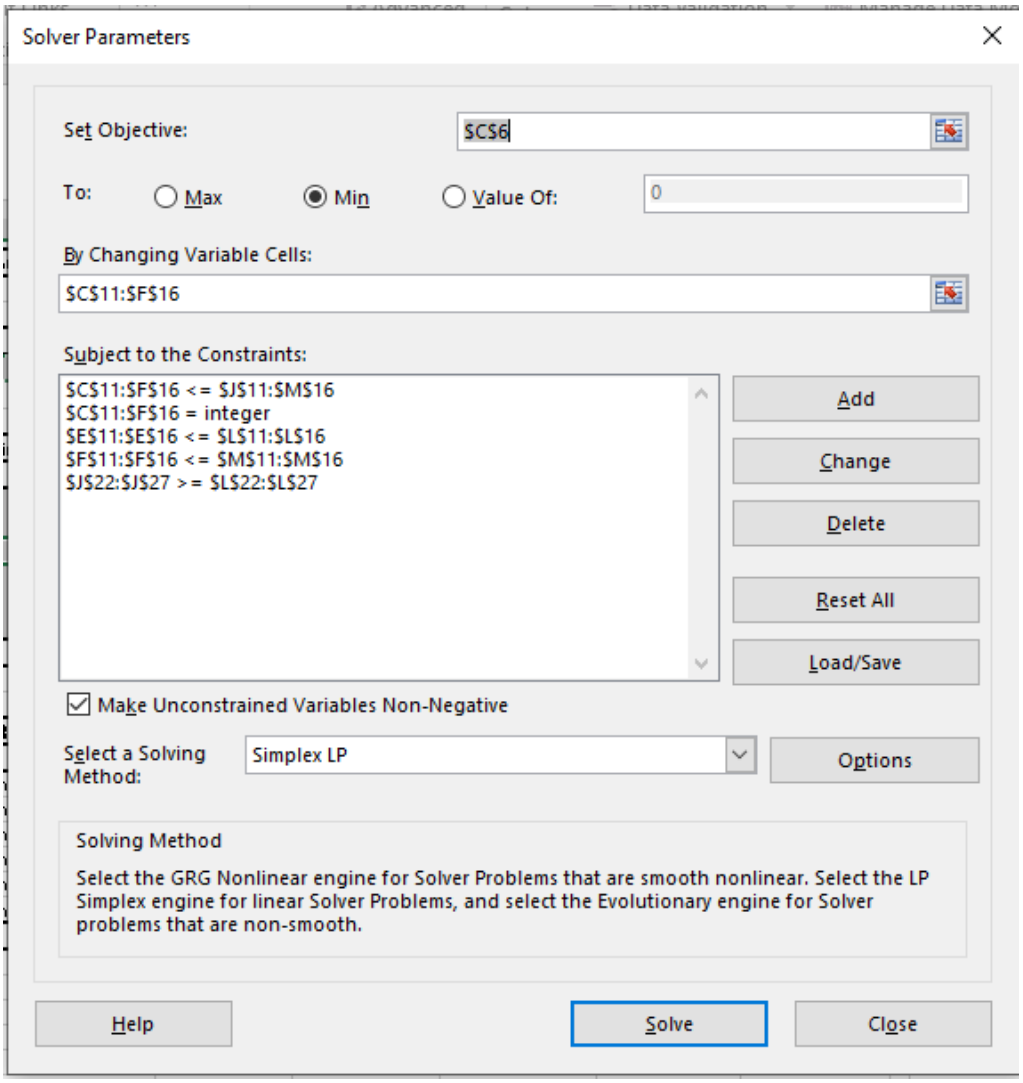


Figure E-3: Model 1 Solver Parameters

Model Goal – This model attempts to show the optimal staffing allocation to minimize the weighted shortage of PCPs b	

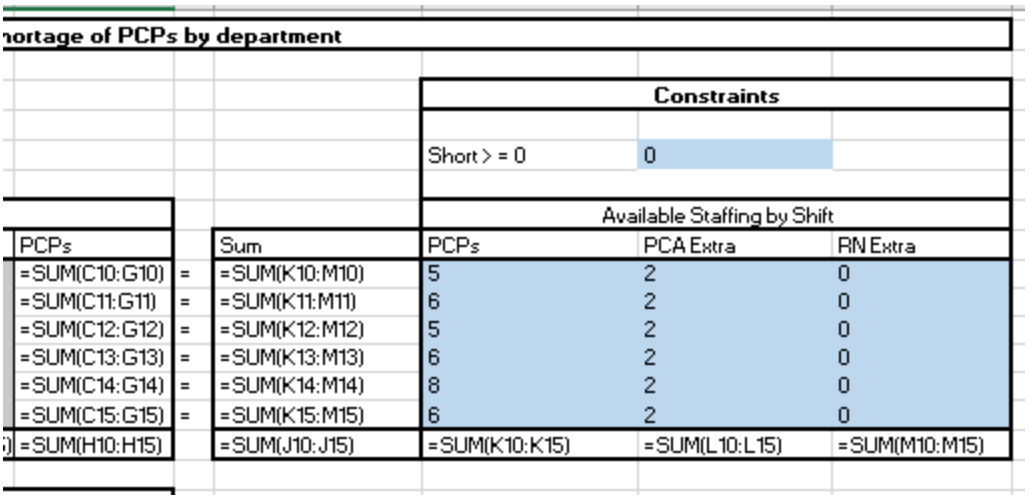
Objective	
Sumsq Weighted Short	=SUMSQ(C20:G
Actual Short	=SUM(C20:G25)

Decisions – Suggested Staffing Allocation						
Block	ED	MS6	MS7	MS8	PICU	PCPs
0700-1100	2	5	0	0	0	=SUM(C10:G10) =
1100-1500	3	5	0	0	0	=SUM(C11:G11) =
1500-1900	1	5	1	0	0	=SUM(C12:G12) =
1900-2300	2	4	1	1	0	=SUM(C13:G13) =
2300-0300	4	2	2	2	0	=SUM(C14:G14) =
0300-0700	4	2	1	1	0	=SUM(C15:G15) =
Sum	=SUM(C10:C15)	=SUM(D10:D15)	=SUM(E10:E15)	=SUM(F10:F15)	=SUM(G10:G15)	=SUM(H10:H15)

# PCPs Short						
Block	ED	MS6	MS7	MS8	PICU	Total
0700-1100	=C30-C10	=D30-D10	=E30-E10	=F30-F10	=G30-G10	=SUM(C20:G20)
1100-1500	=C31-C11	=D31-D11	=E31-E11	=F31-F11	=G31-G11	=SUM(C21:G21)
1500-1900	=C32-C12	=D32-D12	=E32-E12	=F32-F12	=G32-G12	=SUM(C22:G22)
1900-2300	=C33-C13	=D33-D13	=E33-E13	=F33-F13	=G33-G13	=SUM(C23:G23)
2300-0300	=C34-C14	=D34-D14	=E34-E14	=F34-F14	=G34-G14	=SUM(C24:G24)
0300-0700	=C35-C15	=D35-D15	=E35-E15	=F35-F15	=G35-G15	=SUM(C25:G25)
Avg	=SUM(C20:C25)	=SUM(D20:D25)	=SUM(E20:E25)	=SUM(F20:F25)	=SUM(G20:G25	=SUM(C26:G26)

PCP Requirements						
Block	ED	MS6	MS7	MS8	PICU	PCPs
0700-1100	=IFERROR(GETPIVOTDATA("Average of 1:1 Sits Required","Sitter Req'!$B$2,"Dept",C$29,"Timeblock",$B30),0)	=IFERROR(GETF	=IFERROR(GETI	=IFERROR(GET	=IFERROR(GET	=SUM(C30:G30)
1100-1500	=IFERROR(GETPIVOTDATA("Average of 1:1 Sits Required","Sitter Req'!$B$2,"Dept",C$29,"Timeblock",$B31),0)	=IFERROR(GETF	=IFERROR(GETI	=IFERROR(GET	=IFERROR(GET	=SUM(C31:G31)
1500-1900	=IFERROR(GETPIVOTDATA("Average of 1:1 Sits Required","Sitter Req'!$B$2,"Dept",C$29,"Timeblock",$B32),0)	=IFERROR(GETF	=IFERROR(GETI	=IFERROR(GET	=IFERROR(GET	=SUM(C32:G32)
1900-2300	=IFERROR(GETPIVOTDATA("Average of 1:1 Sits Required","Sitter Req'!$B$2,"Dept",C$29,"Timeblock",$B33),0)	=IFERROR(GETF	=IFERROR(GETI	=IFERROR(GET	=IFERROR(GET	=SUM(C33:G33)
2300-0300	=IFERROR(GETPIVOTDATA("Average of 1:1 Sits Required","Sitter Req'!$B$2,"Dept",C$29,"Timeblock",$B34),0)	=IFERROR(GETF	=IFERROR(GETI	=IFERROR(GET	=IFERROR(GET	=SUM(C34:G34)
0300-0700	=IFERROR(GETPIVOTDATA("Average of 1:1 Sits Required","Sitter Req'!$B$2,"Dept",C$29,"Timeblock",$B35),0)	=IFERROR(GETF	=IFERROR(GETI	=IFERROR(GET	=IFERROR(GET	=SUM(C35:G35)
Sum	=SUM(C30:C35)	=SUM(D30:D35)	=SUM(E30:E35)	=SUM(F30:F35)	=SUM(G30:G35	=SUM(H30:H35)

Figure E-4: Model 2 Formulas Displayed – Decisions



hortage of PCPs by department					
			Constraints		
			Short > = 0	0	
			Available Staffing by Shift		
PCPs		Sum	PCPs	PCA Extra	RN Extra
=SUM(C10:G10)	=	=SUM(K10:M10)	5	2	0
=SUM(C11:G11)	=	=SUM(K11:M11)	6	2	0
=SUM(C12:G12)	=	=SUM(K12:M12)	5	2	0
=SUM(C13:G13)	=	=SUM(K13:M13)	6	2	0
=SUM(C14:G14)	=	=SUM(K14:M14)	8	2	0
=SUM(C15:G15)	=	=SUM(K15:M15)	6	2	0
=SUM(H10:H15)		=SUM(J10:J15)	=SUM(K10:K15)	=SUM(L10:L15)	=SUM(M10:M15)

Figure E-5: Model 2 Formulas Displayed – Constraints

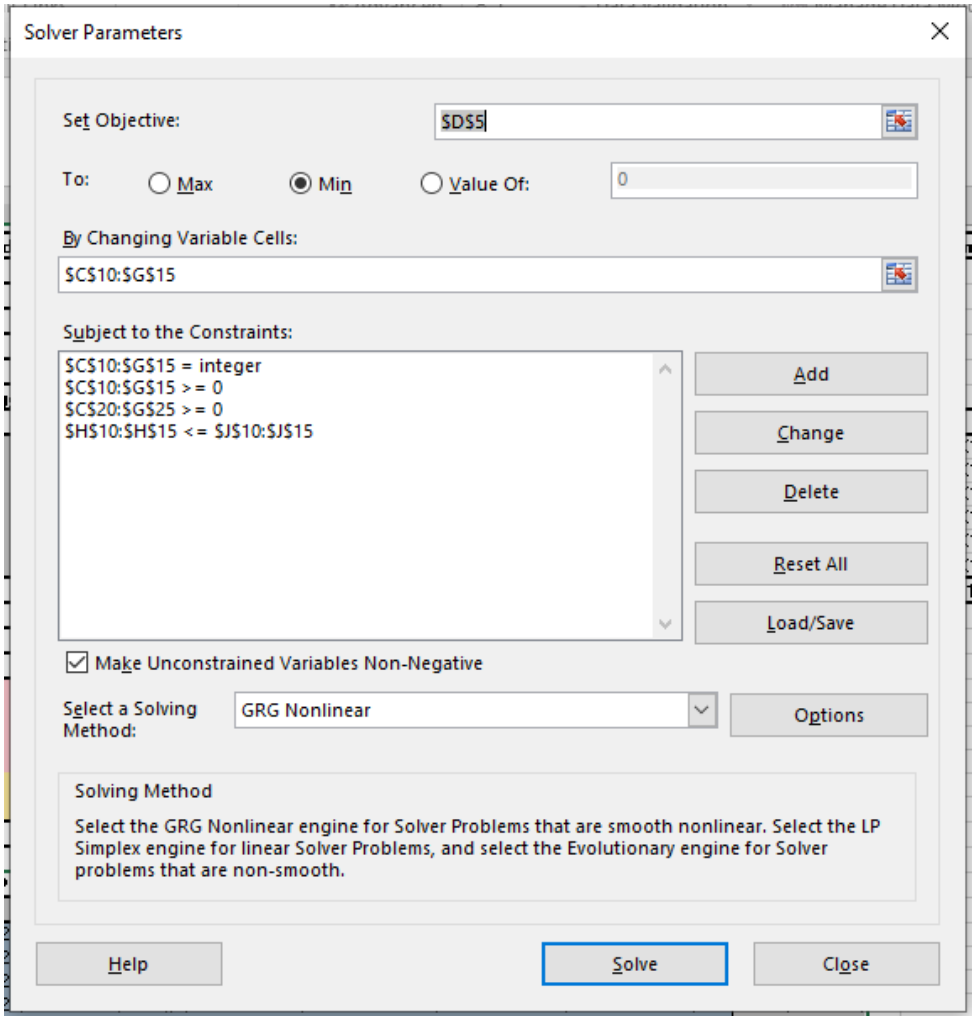


Figure E-6: Model 2 Solver Parameters

7.5 Appendix F – Project Reflection

The design phases of this project were split among three deliverables. The team created an ARIMA forecasting model, defined as a forecasting process improvement, a LIAT model, defined as an inventory forecasting model, and a Behavioral Health Staffing model, defined as an optimization model. Each of these model's opportunities was separately identified. For the ARIMA model, the team was initially tasked with forecasting the impact of a wave two surge of COVID patients. Given the limited information or data to use for this effort, the team identified new opportunities to design a better forecasting system for the metrics CCMC already forecasts and reports. The team identified best practices in the machine learning field and created a forecasting model using the ARIMA methodology. Working closely with experts at CCMC, the

team developed specific requirements for the model and adapted the design process to meet these requirements. Examples of this include building various confidence intervals and retraining the model using grid search parameterization. The team field-tested the model with historical data from CCMC and worked in an agile environment with data analysts at CCMC to receive their constant feedback and adapt to new changes and requirements.

For the LIAT model, opportunities were given to the project team in the form of a specific directed task from the sponsor. In addition to this sponsor request for an inventory model for LIAT tests, the team identified opportunities to use existing modeling practices at CCMC as a baseline for the model and expand on them in a way that met the requirements of determining LIAT inventory burndown rates. Requirements were determined through weekly meetings with the LIAT management and analytics teams at CCMC. Both teams were consulted regularly in a cyclical format to ensure the final deliverables met their expectations. The team also developed a way to test model solutions by tracking actual data compared to the model predictions. Updating the model with live data increased confidence in the model so that experts could safely use it to influence LIAT management decisions.

Finally, the behavioral health optimization modeling opportunities also came from a project sponsor request. This request was a broad one, asking the team to apply their knowledge to help the behavioral health teams within CCMC. The team held several meetings with continuous improvement experts and data analysts to determine what opportunities and requirements existed for a project in the hospital's behavioral health section. These meetings allowed the team to assess our optimization methodology, focusing on staffing of patient care partners (PCPs). The team worked with model end-users and accurate hospital data to verify the model's validity and test different solutions. These meetings were held weekly, and frequent updates to the model were critical to its success as a decision-making tool.

Another critical factor of the project design was considering various constraints. The most prominent of these constraints were the health, safety, economic and political constraints that were introduced because of the COVID-19 pandemic. This pandemic was the catalyst for the project and was a significant factor in the team’s design process in all project phases. For the ARIMA forecasting, the team needed to consider the patient volume impact that COVID had on the hospital’s historical volumes used for modeling and ensure that the model appropriately

responded to unpredictable events. For the LIAT tests, considering the political constraints of rationed LIAT tests and health and safety constraints of achieving proper patient care during a pandemic was very important. For behavioral health, economic constraints on availability to hire staff and turnover rates were essential considerations. Additionally, political constraints in government aid for behavioral health were important considerations to evaluate the model's long term impacts. Overall, the team identified and handled constraints on a case-by-case basis for each model. The success of the models meant accounting for and incorporating all these constraints.

Additionally, the project team learned a vast amount of new information throughout the project. The first grouping of this information was a general understanding of the healthcare industry. Much of the first few weeks of the project were spent trying to understand the healthcare industry's inner workings as it was a new field for all team members. By the end of the project, the team communicated effectively with the CCMC teams and had a workable knowledge of the hospital operations. Additionally, the team learned a significant amount of technical information about ARIMA modeling in Python. While each team member had used Python before, developing an ARIMA model on the scale produced by this project was new territory none of the team had experienced in academic practice. Finally, the team learned a lot about data analytics and continuous improvement in healthcare. The industry has always been focused on meeting patient needs. It is often difficult to gather resources to apply analytics and forecasting in the same way that manufacturing industries can. The team learned a lot about using analytical practices in healthcare, translating to many other industries as they begin to adopt continuous improvement and predictive analytics. These are all skills that, as students, would have been very difficult to learn outside of this capstone project.

Finally, to complete this project, this team had to work together efficiently and effectively. Each team member worked very well with each other. Respect and hard work were never an issue. One of the keys to that success was our organized and systematic approach to communication and project planning. The team used a software program called Trello to manage all our tasks and objectives throughout the project. The software allowed us to have one standing location for all objectives and communicate priorities, due dates, questions, and concerns. The software allowed us to work effectively together in an almost entirely remote setting, a critical

factor in the COVID impacted the world. We also met regularly and communicated openly. Each team member's ability to both provide and take constructive criticism and use it to improve project success is a very commendable attribute. Meeting with the project advisor, project sponsor, and subject matter experts at CCMC frequently allowed the team to communicate challenges and develop solutions in a collaborative and fast-paced environment. All this together allowed the team to perform as an effective, cohesive unit.

References

Boyd, S. P., & Vandenberghe, L. (2004). Convex Optimization. Retrieved January 16, 2021, from https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf#page=143

Brownlee, J. (2020, August 20). SMOTE for Imbalanced Classification with Python. Retrieved from https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

Brownlee, J. (2020, December 09). How to create an arima model for time series forecasting in python. Retrieved from https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/

Connecticut children's. (n.d.). Retrieved from https://www.connecticutchildrens.org/

Coronavirus. (2020, February 15). Retrieved October 08, 2020, from https://www.cdc.gov/coronavirus/types.html

Da'ar, O. B., & Ahmed, A. E. (2018). Underlying trend, seasonality, prediction, forecasting and the contribution of risk factors: an analysis of globally reported cases of Middle East Respiratory Syndrome Coronavirus. *Epidemiology and infection*, *146*(11), 1343–1349. https://doi.org/10.1017/S0950268818001541

Define and solve a problem by using solver. (n.d.). Retrieved January 16, 2021, from https://support.microsoft.com/en-us/office/define-and-solve-a-problem-by-using-solver-5d1a388f-079d-43ac-a7eb-f63e45925040

Mackay, M. (2005, June). Modelling Variability in Hospital Bed Occupancy. Retrieved from https://link.springer.com/content/pdf/10.1007/s10729-005-4142-8.pdf

Nau, R. (2014, December 13). The mathematical structure of ARIMA models. Retrieved from http://people.duke.edu/~rnau/Mathematical_structure_of_ARIMA_models--Robert_Nau.pdf

NOAA National Centers for Environmental Information State Climate Summaries. (n.d.). Retrieved October 08, 2020, from https://statesummaries.ncics.org/chapter/ct/

Pica, N., & Bouvier, N. M. (2012). Environmental factors affecting the transmission of respiratory viruses. *Current Opinion in Virology*, *2*(1), 90-95. doi:10.1016/j.coviro.2011.12.003

Pilgrim, T. (2020, August 06). Localised coronavirus simulations predict second wave in 'almost all cases'. Retrieved October, from https://medicalxpress.com/news/2020-08-localised-coronavirus-simulations-cases.html

Schooling, C. N., Gyenge, N., Kadirkamanathan, V., & Alix, J. J. (2020, April 17). Forecasting Hospital Staff Availability During The COVID-19 Epidemic. Retrieved from https://www.medrxiv.org/content/10.1101/2020.04.15.20066019v1.full.pdf

3-Step Process for Predictive Analytics Forecasting. (2020, April 27). Retrieved October 09, 2020, from https://www.zencos.com/blog/advanced-analytics-model-guide-forecasting/

Weather. (n.d.). Retrieved October 08, 2020, from https://www.universaltraveller.com.au/destinations/middle-east/weather