Team Medecipher Final Report

12 Jun 2020

Connor Graflund
Frederick Habelko
Amy Hong
Liam Stacy
William Struck

I. Introduction

Medecipher is a Denver-based health IT start-up that aims to optimize clinical operations. They provide hospitals with forecasts of the expected number of patients in a given hour so that the hospital can appropriately allocate its resources. Medecipher's current forecasting model, ARIMAX, helps their hospitals to better match the number of needed nurses to the volume of patients.

The goal of this project was to manage nurse staffing in the emergency environment more efficiently. In hospitals, there is currently a lot of variability in the input and output of patients. For this reason, it is difficult to predict the required amount of staffing for any given time. Using SageMaker, our first task was to create a more accurate and precise predictive model than the one Medecipher is currently using, working with years of hospital data. Nurses have variable-length shifts (8 hours, 10 hours, or 12 hours), and the rule-of-thumb in hospitals is to maintain a 4:1 ratio for patients to nurses. With this information, we created different suggestive nurse schedules, differentiated by how much the 4:1 ratio is violated by the model (20%, 15%, or 5% ratio violation).

Besides statistical modeling, we also worked on the front-end user interface of the Medecipher website. The current state of the Medecipher website allows for hospitals to select one given day for a few different staffing schedules. To better handle the problem, we designed a six-week calendar for clients to use, with color coding and possible alterations for the client to use. We also wanted the client (hospitals) to be able to filter specifications for what they would like in their schedule.

II. Functional Requirements

Task 1 [Improve the Forecasting Model]:

1. Set up AWS workflow to build & deploy models using SageMaker - Jupyter Notebooks
2. Understand current ARIMAX forecasting model parameters, input, accuracy, and challenges
3. Build models for producing hourly census & arrival forecasts using SageMaker
    a. Models to be built: Prophet, LSTM, N-Beats, Simple Feed Forward, DeepAR
4. Publish code to AWS Code Commit using Jupyter notebook workflow
5. Create a GUI to input parameter values
6. Produce control charts for error monitoring

Task 2 [User Interface/Front-end]:

1. Create angular component that displays a schedule planner based on the forecasting model from Task 1
2. Features:
    a. Filters: Fiscal Year (FY), block, day of week
    b. Printing: Ability to print the daily schedule (4a today thru 11a tomorrow)
3. Data interface
    a. Incorporate static, hard-coded data to the front-end using JSON or other data objects.

III. Non-Functional Requirements

1. The code for the forecasting model must be written in Python, R, or AMPL.
2. The program must use the Jupyter Notebook development environment.
3. The forecasting model must be built using AWS SageMaker.
4. UI mockups must have a comprehensive workflow.

IV. System Architecture and Technical Design

Task 1 (Improve the Forecasting Model):

DeepAR is a machine learning algorithm built into Amazon Sagemaker. DeepAR uses Recurrent Neural Networks to make both point and probabilistic predictions. Training data is inputted as a JSON file and prediction requests are made to an endpoint using JSON lines commands. The workflow for implementing the DeepAR model is shown below in Figure 1.

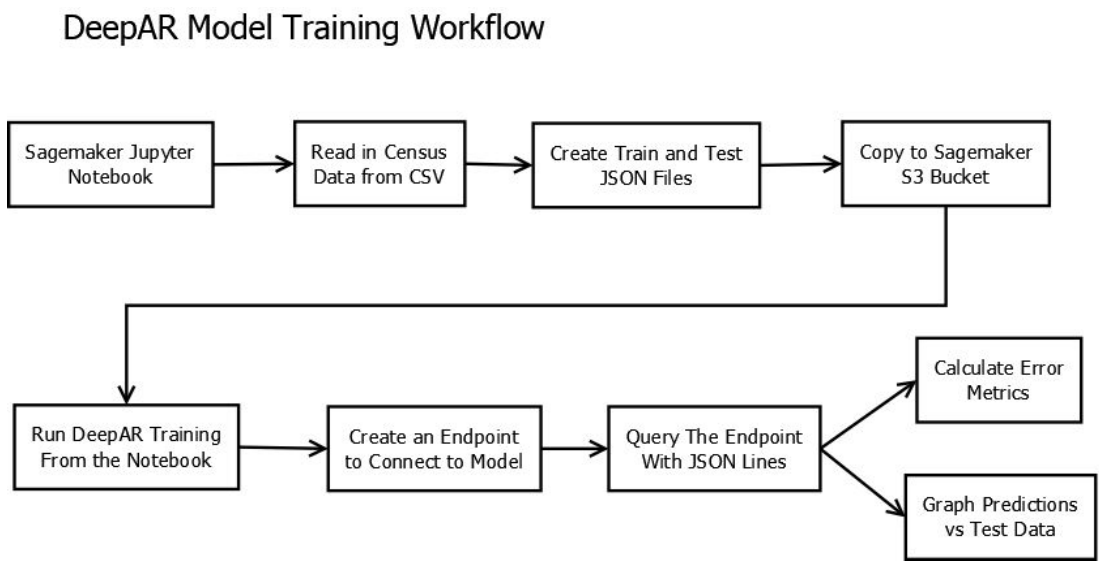

DeepAR Model Training Workflow
Sagemaker Jupyter Notebook
Read in Census Data from CSV
Create Train and Test JSON Files
Copy to Sagemaker S3 Bucket
Run DeepAR Training From the Notebook
Create an Endpoint to Connect to Model
Query The Endpoint With JSON Lines
Calculate Error Metrics
Graph Predictions vs Test Data

Figure 1: DeepAR workflow

Figure 2 below is the workflow for the Simple Feed Forward and N-Beats models. Both these models are built into the GluonTS toolkit for time series modeling, and therefore have similar workflows. Census data is read in from a CSV file and split up into training and testing data; these sets of data are then converted into ListDataSets objects so that they are in the correct format to make predictions. An estimator is created with all the hyperparameters for the model; this is the only part of the code that is different for the two models as each model has its own defined estimator class. A predictor is created from the estimator and is used to make forecasts.

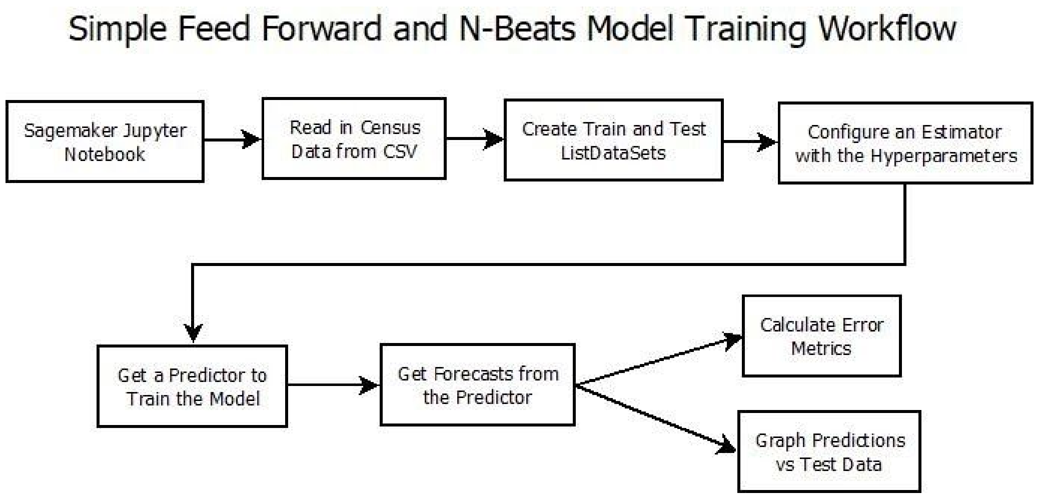

Simple Feed Forward and N-Beats Model Training Workflow
Sagemaker Jupyter Notebook
Read in Census Data from CSV
Create Train and Test ListDataSets
Configure an Estimator with the Hyperparameters
Get a Predictor to Train the Model
Get Forecasts from the Predictor
Calculate Error Metrics
Graph Predictions vs Test Data

Figure 2: Simple Feed Forward and N-Beats Workflow

The Prophet model was developed by Facebook to predict website traffic. It uses Fourier analysis to fit seasonal (periodic) data. In addition there are other interesting features that the model can use in it's predictions including: standard holiday's from the country in which the data is taken from, weekly seasonality, specific Fourier orders for each regressor etc. In this application we are using Prophet to predict hospital ER patient arrivals. Figure 3 Below shows the workflow for Prophet.

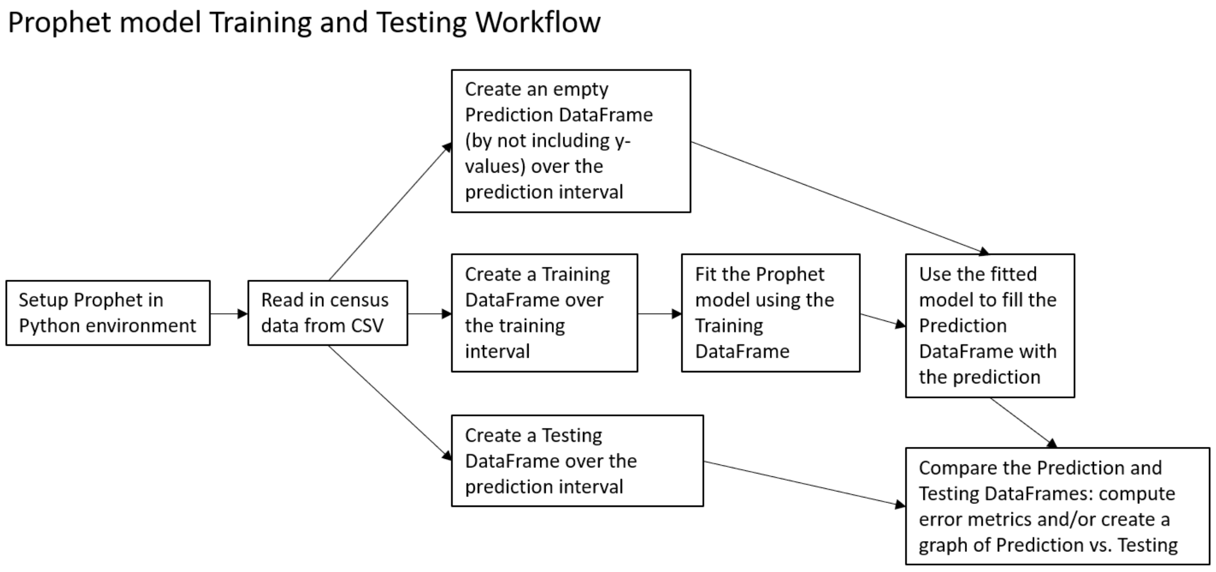
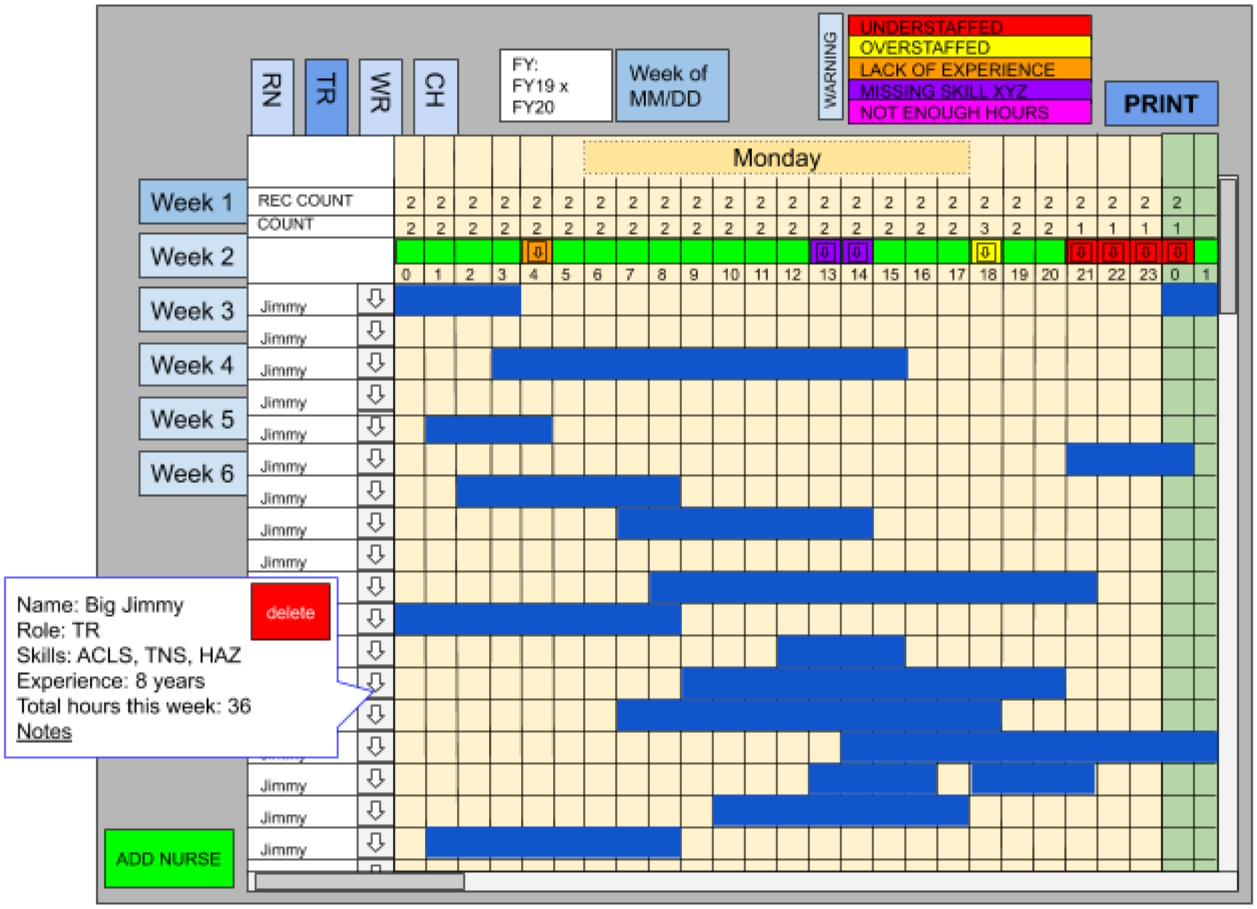

Prophet model Training and Testing Workflow
Setup Prophet in Python environment
Read in census data from CSV
Create an empty Prediction DataFrame (by not including y-values) over the prediction interval
Create a Training DataFrame over the training interval
Create a Testing DataFrame over the prediction interval
Fit the Prophet model using the Training DataFrame
Use the fitted model to fill the Prediction DataFrame with the prediction
Compare the Prediction and Testing DataFrames: compute error metrics and/or create a graph of Prediction vs. Testing

Figure 3: Prophet Workflow

Task 2 (User Interface/Front-end):

Ultimately, the front end of the project will be created by Medecipher in AngularJS. The website will be connected to the rest of the website via a python application, such as Flask, to update the schedule planner and nursing forecasting model in real time. Each different component of the website will have its own corresponding type script, CSS and HTML so that everything is styled appropriately and located where it should be on the site. As of right now, we have created mock up models of potential designs to pursue with the website.

Final Website Mockup



Name: Big Jimmy
Role: TR
Skills: ACLS, TNS, HAZ
Experience: 8 years
Total hours this week: 36
Notes

Model Description:
Schedule module:

This model is based on a more interactive spreadsheet style planner. It also is based on the scheduling software used at golf courses.

Nurses Column:

Each nurse has their own row. Clicking on the [↓] in the nurse row provides information on the selected nurse. To register a nurse for a specific time, the user needs to just click on a box in the grid, and the nurse will be scheduled for that time. Clicking on a box with a shift already scheduled removes the shift for that hour.

Warning Row:

The colored row above the time is the warning indicator row. If there is no warning, it will simply light up as a green box. If there is a warning, the respective column for the hour will change color to the most severe warning and create a pulldown menu ([↓]) containing a description of all the warnings for a given hour.

<u>Role Selector:</u>

This selects which role to display the schedule for.

<u>Week Selector:</u>

This selects which week to display the schedule for.

<u>Print Button:</u>

Brings the user to the print screen.

<u>Add Nurse Button:</u>

The “Add Nurse” button creates a popup that will ask for information on the nurse being added. After being added, a nurse will appear in a new role for the role(s) they were selected for. The other information can be accessed from the [↓] in the respective nurse’s row.

Limitations:

- Shift must be punched in hour by hour.
- Potential display issue from the end of a week to the beginning of the next.
- Removing a nurse from one role could end up removing them from all roles. If not, reregistering that nurse could create a duplicate.
- Roles are in separate views and may be difficult to compare with each other.

Possible Small Changes:

- Add a way to schedule a shift from a Start to Finish.
- Connecting to a database that contains “Nurse” objects would make “ADD NURSE” much easier.
- Make it obvious how many nurses are needed at any given hour so the manager doesn’t have to guess and get a warning first.

V. Quality Assurance

Task 1 (Back End):

- Code review
    - During one meeting, we determined that the LSTM model was proving too difficult to implement, due to lack of documentation and little versatility, so we moved on to other algorithms such as simple feed forward and N-Beats.
- User acceptance testing
    - Error metrics of new model should beat error metrics of previous model and work well over time.
    - Met with Medecipher and ran through Jupyter notebooks for validation.
    - Each of us on the data science side met with Medeciphers data science specialist, Kevin, one-on-one to ensure the quality of our jupyter notebooks. All of us were told that the notebook code and documentation is acceptable for their use.

Task 2 (User Interface/Front-end):

- Design Mock-ups for review before implementing design
    - A discussion and/or wireframe was thought through before the next stage of the UI is implemented.
- Weekly review meetings with Parker from Medecipher
    - Every Thursday, we met with the main connection between us and Medecipher, Parker, to ensure that we understood the task at hand and how to proceed with the project.
    - Meetings with Medecipher occur 2-3 times a week in order to make sure priorities stay in line with expectations. Every other week there is a larger scale meeting with Medecipher in which Task 1 and Task 2 work correlate with 10+ members of Medecipher to produce a plan for the following weeks.
    - Within these meetings, we also discussed the requirements of the final project, as they changed on more than one occasion.

VI. Results

Task 1 (Forecasting/Back End):

Throughout trying to improve the forecasting algorithm, we experimented with 5 different models: DeepAR, Prophet, Simple Feed Forward, N-Beats, and LSTM. The ultimate goal in each of these models is to minimize the desired error metrics mean absolute percentage error (MAPE), root mean square error (RMSE), and mean absolute error (MAE), and to reach better error metrics than the current model Medecipher is using, called ARIMAX.

DeepAR

Figure 4 shows the graph of a six-week forecast from DeepAR versus the test data. The DeepAR model has slightly worse error metrics than the ARIMAX model. This could be improved with more hyperparameter tuning. Because DeepAR takes a long time to train models and requires creating and using an endpoint, it is more time-consuming to tune than other models.

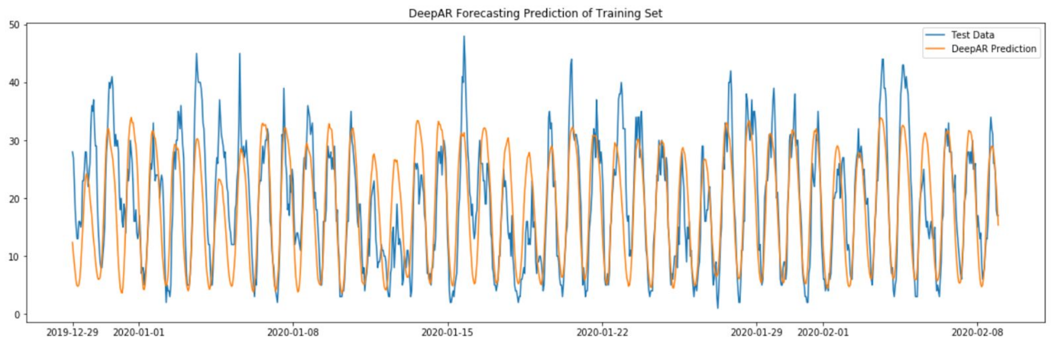


Figure 4: DeepAR Prediction vs. Test Data

After creating the error monitoring chart in Figure 5, we decided to try averaging the models together as shown in Figure 6 to see if it is more accurate than Arimax. The error metrics for the combined model were slightly better than both Arimax and DeepAR. It could be worth exploring combining the models to improve accuracy.

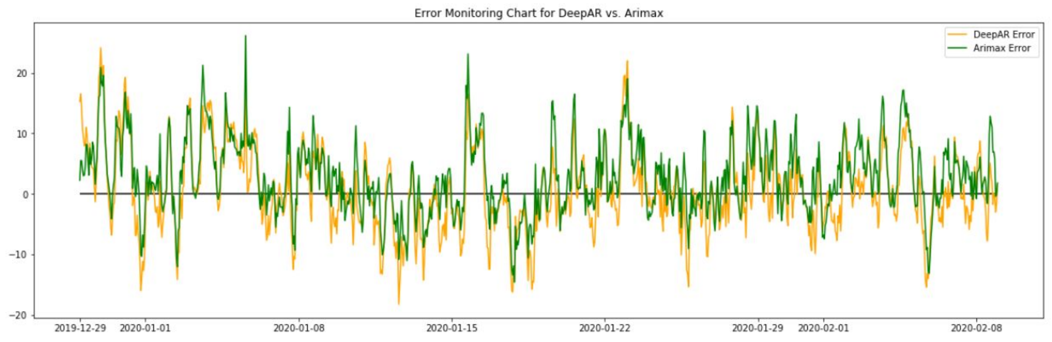


Figure 5: Error monitoring chart for DeepAR vs. Arimax

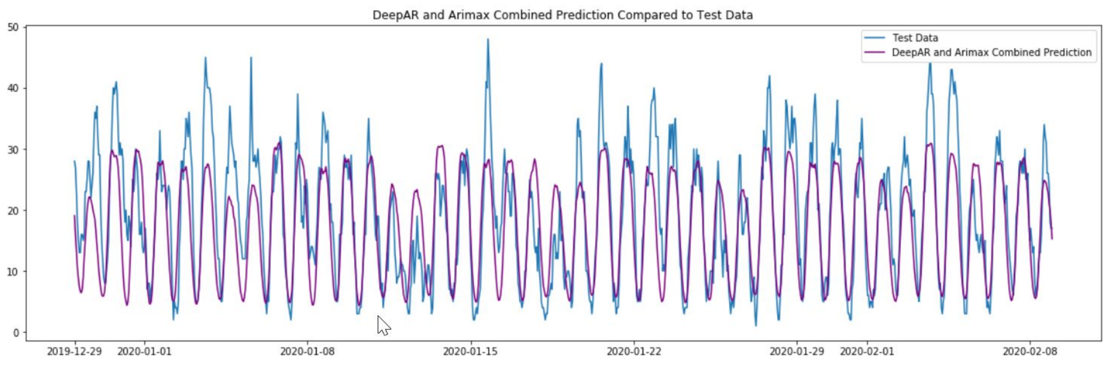
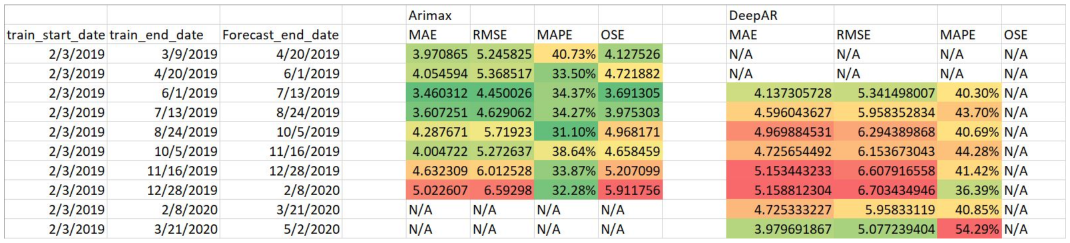


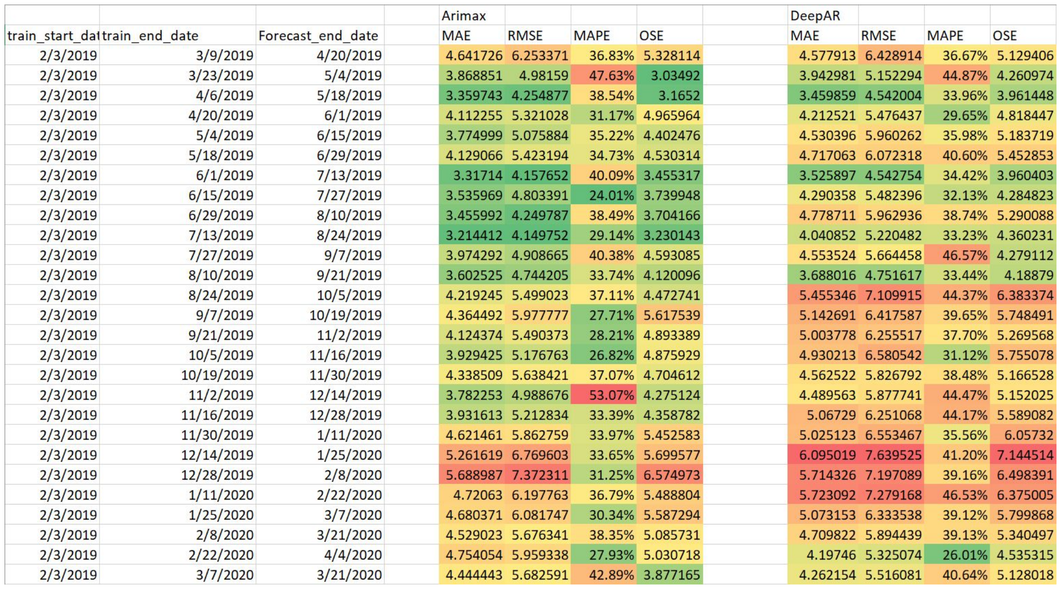
Figure 6: DeepAR and Arimax Combined Average Prediction

Figures 7 and 8 show the error metrics for DeepAR compared to Arimax for a six and two week forecast respectively. Due to time constraints, long training time, and model limitations the metrics for the combined model and some other data are missing. DeepAR did not perform better than Arimax over a one-year cycle.



			Arimax				DeepAR			
train_start_date	train_end_date	Forecast_end_date	MAE	RMSE	MAPE	OSE	MAE	RMSE	MAPE	OSE
2/3/2019	3/9/2019	4/20/2019	3.970865	5.245825	40.73%	4.127526	N/A	N/A	N/A	N/A
2/3/2019	4/20/2019	6/1/2019	4.054594	5.368517	33.50%	4.721882	N/A	N/A	N/A	N/A
2/3/2019	6/1/2019	7/13/2019	3.460312	4.450026	34.37%	3.691305	4.137305728	5.341498007	40.30%	N/A
2/3/2019	7/13/2019	8/24/2019	3.607251	4.629062	34.27%	3.975303	4.596043627	5.958352834	43.70%	N/A
2/3/2019	8/24/2019	10/5/2019	4.287671	5.71923	31.10%	4.968171	4.969884531	6.294389868	40.69%	N/A
2/3/2019	10/5/2019	11/16/2019	4.004722	5.272637	38.64%	4.658459	4.725654492	6.153673043	44.28%	N/A
2/3/2019	11/16/2019	12/28/2019	4.632309	6.012528	33.87%	5.207099	5.153443233	6.607916558	41.42%	N/A
2/3/2019	12/28/2019	2/8/2020	5.022607	6.59298	32.28%	5.911756	5.158812304	6.703434946	36.39%	N/A
2/3/2019	2/8/2020	3/21/2020	N/A	N/A	N/A	N/A	4.725333227	5.95833119	40.85%	N/A
2/3/2019	3/21/2020	5/2/2020	N/A	N/A	N/A	N/A	3.979691867	5.077239404	54.29%	N/A

Figure 7: DeepAR vs. Arimax 6-week Error Metrics



			Arimax				DeepAR			
train_start_dat	train_end_date	Forecast_end_date	MAE	RMSE	MAPE	OSE	MAE	RMSE	MAPE	OSE
2/3/2019	3/9/2019	4/20/2019	4.641726	6.253371	36.83%	5.328114	4.577913	6.428914	36.67%	5.129406
2/3/2019	3/23/2019	5/4/2019	3.868851	4.98159	47.63%	3.03492	3.942981	5.152294	44.87%	4.260974
2/3/2019	4/6/2019	5/18/2019	3.359743	4.254877	38.54%	3.1652	3.459859	4.542004	33.96%	3.961448
2/3/2019	4/20/2019	6/1/2019	4.112255	5.321028	31.17%	4.965964	4.212521	5.476437	29.65%	4.818447
2/3/2019	5/4/2019	6/15/2019	3.774999	5.075884	35.22%	4.402476	4.530396	5.960262	35.98%	5.183719
2/3/2019	5/18/2019	6/29/2019	4.129066	5.423194	34.73%	4.530314	4.717063	6.072318	40.60%	5.452853
2/3/2019	6/1/2019	7/13/2019	3.31714	4.157652	40.09%	3.455317	3.525897	4.542754	34.42%	3.960403
2/3/2019	6/15/2019	7/27/2019	3.535969	4.803391	24.01%	3.739948	4.290358	5.482396	32.13%	4.284823
2/3/2019	6/29/2019	8/10/2019	3.455992	4.249787	38.49%	3.704166	4.778711	5.962936	38.74%	5.290088
2/3/2019	7/13/2019	8/24/2019	3.214412	4.149752	29.14%	3.230143	4.040852	5.220482	33.23%	4.360231
2/3/2019	7/27/2019	9/7/2019	3.974292	4.908665	40.38%	4.593085	4.553524	5.664458	46.57%	4.279112
2/3/2019	8/10/2019	9/21/2019	3.602525	4.744205	33.74%	4.120096	3.688016	4.751617	33.44%	4.18879
2/3/2019	8/24/2019	10/5/2019	4.219245	5.499023	37.11%	4.472741	5.455346	7.109915	44.37%	6.383374
2/3/2019	9/7/2019	10/19/2019	4.364492	5.977777	27.71%	5.617539	5.142691	6.417587	39.65%	5.748491
2/3/2019	9/21/2019	11/2/2019	4.124374	5.490373	28.21%	4.893389	5.003778	6.255517	37.70%	5.269568
2/3/2019	10/5/2019	11/16/2019	3.929425	5.176763	26.82%	4.875929	4.930213	6.580542	31.12%	5.755078
2/3/2019	10/19/2019	11/30/2019	4.338509	5.638421	37.07%	4.704612	4.562522	5.826792	38.48%	5.166528
2/3/2019	11/2/2019	12/14/2019	3.782253	4.988676	53.07%	4.275124	4.489563	5.877741	44.47%	5.152025
2/3/2019	11/16/2019	12/28/2019	3.931613	5.212834	33.39%	4.358782	5.06729	6.251068	44.17%	5.589082
2/3/2019	11/30/2019	1/11/2020	4.621461	5.862759	33.97%	5.452583	5.025123	6.553467	35.56%	6.05732
2/3/2019	12/14/2019	1/25/2020	5.261619	6.769603	33.65%	5.699577	6.095019	7.639525	41.20%	7.144514
2/3/2019	12/28/2019	2/8/2020	5.688987	7.372311	31.25%	6.574973	5.714326	7.197089	39.16%	6.498391
2/3/2019	1/11/2020	2/22/2020	4.72063	6.197763	36.79%	5.488804	5.723092	7.279168	46.53%	6.375005
2/3/2019	1/25/2020	3/7/2020	4.680371	6.081747	30.34%	5.587294	5.073153	6.333538	39.12%	5.799868
2/3/2019	2/8/2020	3/21/2020	4.529023	5.676341	38.35%	5.085731	4.709822	5.894439	39.13%	5.340497
2/3/2019	2/22/2020	4/4/2020	4.754054	5.959338	27.93%	5.030718	4.19746	5.325074	26.01%	4.535315
2/3/2019	3/7/2020	3/21/2020	4.444443	5.682591	42.89%	3.877165	4.262154	5.516081	40.64%	5.128018

Figure 8: DeepAR vs. Arimax 2-week Error Metrics

Prophet

Figure 9 below shows an example of a Prophet forecast (shown in orange) in comparison to the actual number of ER patients (shown in blue) for a six week period.

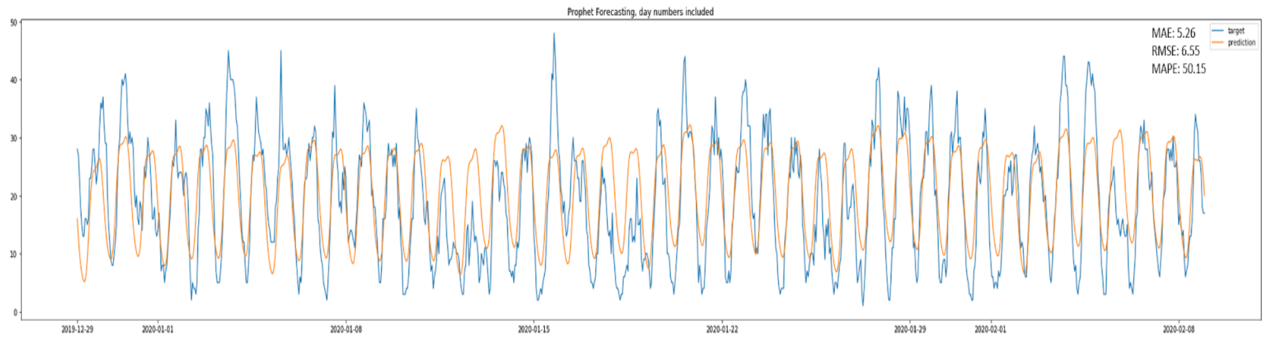
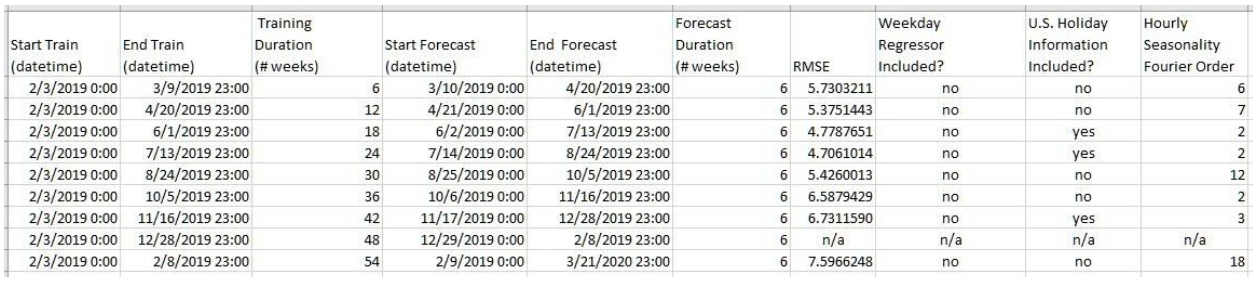


Figure 9: Example Prophet Prediction vs Test Data

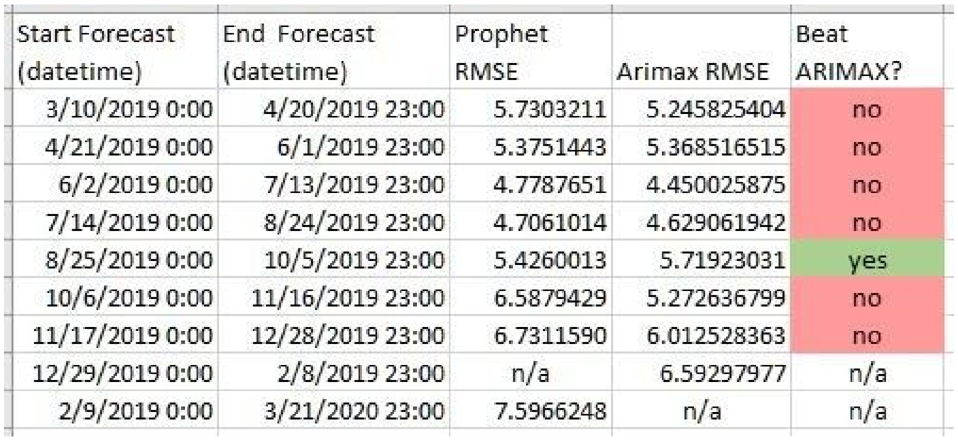
A benefit of the Prophet model is that it uses Fourier transforms to fit each model, instead of gradient descent like other machine learning methods; this means that it doesn't take very long to train a model for testing. Because of this convenient fact many different possible combinations of hyperparameters could be tested to see which ones performed best. Overall, 540 different combinations of hyperparameters were tested. Error metrics for each 6-week testing period were collected for all 540 combinations of the following hyperparameters: 9 different training period lengths, 20 different hourly Fourier orders, whether or not a day-of-the-week integer was included, and whether or not U.S. holiday data was included. Figure 10 below shows the best possible combination of hyperparameters for each 6-week period.



Start Train (datetime)	End Train (datetime)	Training Duration (# weeks)	Start Forecast (datetime)	End Forecast (datetime)	Forecast Duration (# weeks)	RMSE	Weekday Regressor Included?	U.S. Holiday Information Included?	Hourly Seasonality Fourier Order
2/3/2019 0:00	3/9/2019 23:00	6	3/10/2019 0:00	4/20/2019 23:00	6	5.7303211	no	no	6
2/3/2019 0:00	4/20/2019 23:00	12	4/21/2019 0:00	6/1/2019 23:00	6	5.3751443	no	no	7
2/3/2019 0:00	6/1/2019 23:00	18	6/2/2019 0:00	7/13/2019 23:00	6	4.7787651	no	yes	2
2/3/2019 0:00	7/13/2019 23:00	24	7/14/2019 0:00	8/24/2019 23:00	6	4.7061014	no	yes	2
2/3/2019 0:00	8/24/2019 23:00	30	8/25/2019 0:00	10/5/2019 23:00	6	5.4260013	no	no	12
2/3/2019 0:00	10/5/2019 23:00	36	10/6/2019 0:00	11/16/2019 23:00	6	6.5879429	no	no	2
2/3/2019 0:00	11/16/2019 23:00	42	11/17/2019 0:00	12/28/2019 23:00	6	6.7311590	no	yes	3
2/3/2019 0:00	12/28/2019 23:00	48	12/29/2019 0:00	2/8/2019 23:00	6	n/a	n/a	n/a	n/a
2/3/2019 0:00	2/8/2019 23:00	54	2/9/2019 0:00	3/21/2020 23:00	6	7.5966248	no	no	18

Figure 10: The best RMSE value for each combination of hyperparameters using the Prophet model

Only one of the 6-week period's RMSE value outperformed ARIMAX as shown in Figure 11 below:



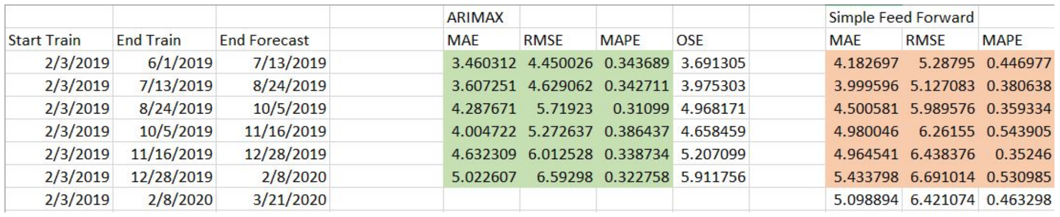
Start Forecast (datetime)	End Forecast (datetime)	Prophet RMSE	Arimax RMSE	Beat ARIMAX?
3/10/2019 0:00	4/20/2019 23:00	5.7303211	5.245825404	no
4/21/2019 0:00	6/1/2019 23:00	5.3751443	5.368516515	no
6/2/2019 0:00	7/13/2019 23:00	4.7787651	4.450025875	no
7/14/2019 0:00	8/24/2019 23:00	4.7061014	4.629061942	no
8/25/2019 0:00	10/5/2019 23:00	5.4260013	5.71923031	yes
10/6/2019 0:00	11/16/2019 23:00	6.5879429	5.272636799	no
11/17/2019 0:00	12/28/2019 23:00	6.7311590	6.012528363	no
12/29/2019 0:00	2/8/2019 23:00	n/a	6.59297977	n/a
2/9/2019 0:00	3/21/2020 23:00	7.5966248	n/a	n/a

Figure 11: The best RMSE value for each 6-week testing interval: Prophet vs Arimax

Thus, even when fully tuned, the Prophet model rarely beats ARIMAX. The conclusion is that Prophet is not a better model than ARIMAX for this application.

Simple Feed Forward

The client's current model outperforms this model in terms of all three reported error metrics; therefore, this model does not meet our client's functional requirements. Figure 12 shows the error metrics for six-week forecasts using the Simple Feed Forward model and compares it to the error metrics from ARIMAX. The green indicates that a model produces better error metrics than the other for that training period, while red indicates the model performs more poorly.



				ARIMAX					Simple Feed Forward		
Start Train	End Train	End Forecast		MAE	RMSE	MAPE	OSE		MAE	RMSE	MAPE
2/3/2019	6/1/2019	7/13/2019		3.460312	4.450026	0.343689	3.691305		4.182697	5.28795	0.446977
2/3/2019	7/13/2019	8/24/2019		3.607251	4.629062	0.342711	3.975303		3.999596	5.127083	0.380638
2/3/2019	8/24/2019	10/5/2019		4.287671	5.71923	0.31099	4.968171		4.500581	5.989576	0.359334
2/3/2019	10/5/2019	11/16/2019		4.004722	5.272637	0.386437	4.658459		4.980046	6.26155	0.543905
2/3/2019	11/16/2019	12/28/2019		4.632309	6.012528	0.338734	5.207099		4.964541	6.438376	0.35246
2/3/2019	12/28/2019	2/8/2020		5.022607	6.59298	0.322758	5.911756		5.433798	6.691014	0.530985
2/3/2019	2/8/2020	3/21/2020							5.098894	6.421074	0.463298

Figure 12: Simple Feed Forward vs. ARIMAX 6-week Error Metrics

Due to time constraints, two-week forecasts over a one year period were not produced for the Simple Feed Forward model.

N-Beats

This model produces slightly worse error metrics than the ARIMAX model for some of the six-week blocks. Given more time to tune the hyperparameters for specific blocks in the fiscal year, this model could produce improved metrics. Figure 13 shows the graph of a six-week forecast from N-Beats and the test data.

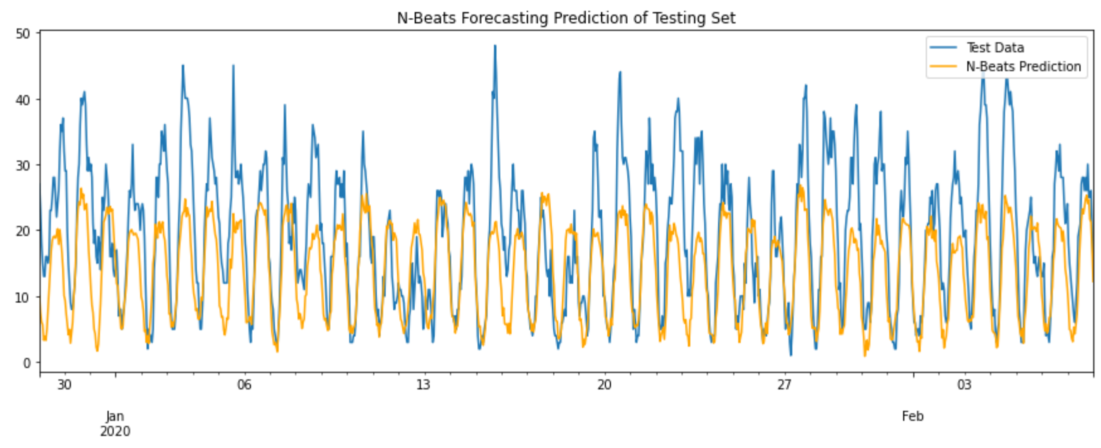
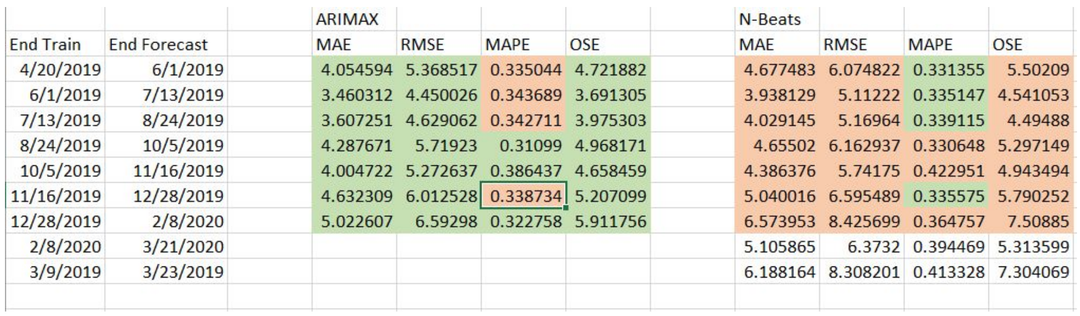


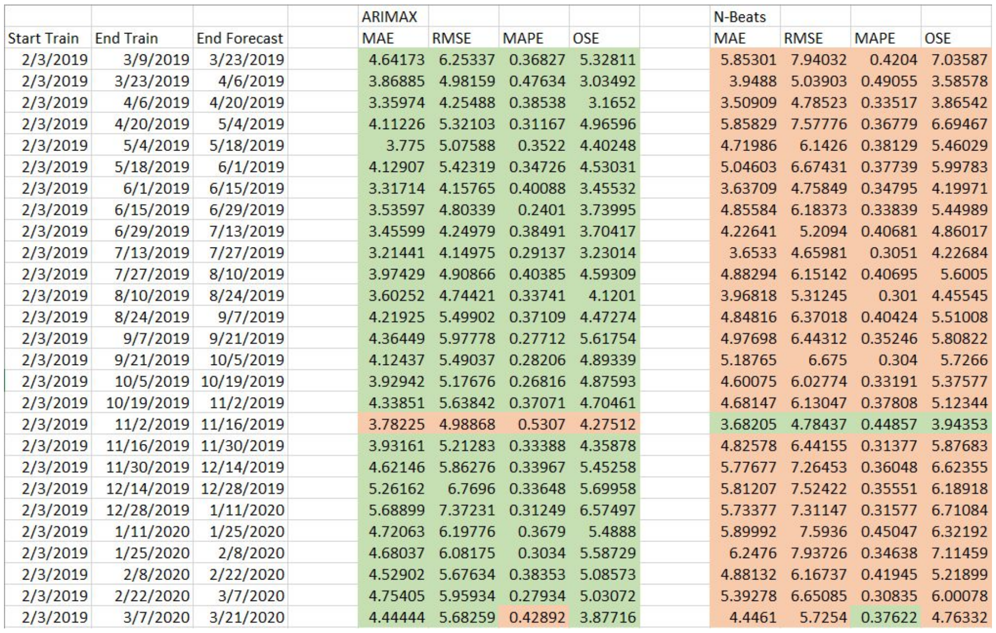
Figure 13: N-Beats Prediction vs Test Data

N-Beats was used to run both two-week and six-week forecasts. Figures 14 and 15 show the error metrics for each forecast length compared to the error metrics for ARIMAX. These figures are color-coded such that the model with the better error metric for that training period is highlighted green, and the model with the worse error metric is highlighted red. From Figures 14 and 15, it is clear that ARIMAX outperforms N-Beats for both the two-week and six-week forecasts over a one year cycle.



		ARIMAX				N-Beats			
End Train	End Forecast	MAE	RMSE	MAPE	OSE	MAE	RMSE	MAPE	OSE
4/20/2019	6/1/2019	4.054594	5.368517	0.335044	4.721882	4.677483	6.074822	0.331355	5.50209
6/1/2019	7/13/2019	3.460312	4.450026	0.343689	3.691305	3.938129	5.11222	0.335147	4.541053
7/13/2019	8/24/2019	3.607251	4.629062	0.342711	3.975303	4.029145	5.16964	0.339115	4.49488
8/24/2019	10/5/2019	4.287671	5.71923	0.31099	4.968171	4.65502	6.162937	0.330648	5.297149
10/5/2019	11/16/2019	4.004722	5.272637	0.386437	4.658459	4.386376	5.74175	0.422951	4.943494
11/16/2019	12/28/2019	4.632309	6.012528	0.338734	5.207099	5.040016	6.595489	0.335575	5.790252
12/28/2019	2/8/2020	5.022607	6.59298	0.322758	5.911756	6.573953	8.425699	0.364757	7.50885
2/8/2020	3/21/2020					5.105865	6.3732	0.394469	5.313599
3/9/2019	3/23/2019					6.188164	8.308201	0.413328	7.304069

Figure 14: N-Beats vs. ARIMAX 6-week Error Metrics



				ARIMAX					N-Beats			
Start Train	End Train	End Forecast		MAE	RMSE	MAPE	OSE		MAE	RMSE	MAPE	OSE
2/3/2019	3/9/2019	3/23/2019		4.64173	6.25337	0.36827	5.32811		5.85301	7.94032	0.4204	7.03587
2/3/2019	3/23/2019	4/6/2019		3.86885	4.98159	0.47634	3.03492		3.9488	5.03903	0.49055	3.58578
2/3/2019	4/6/2019	4/20/2019		3.35974	4.25488	0.38538	3.1652		3.50909	4.78523	0.33517	3.86542
2/3/2019	4/20/2019	5/4/2019		4.11226	5.32103	0.31167	4.96596		5.85829	7.57776	0.36779	6.69467
2/3/2019	5/4/2019	5/18/2019		3.775	5.07588	0.3522	4.40248		4.71986	6.1426	0.38129	5.46029
2/3/2019	5/18/2019	6/1/2019		4.12907	5.42319	0.34726	4.53031		5.04603	6.67431	0.37739	5.99783
2/3/2019	6/1/2019	6/15/2019		3.31714	4.15765	0.40088	3.45532		3.63709	4.75849	0.34795	4.19971
2/3/2019	6/15/2019	6/29/2019		3.53597	4.80339	0.2401	3.73995		4.85584	6.18373	0.33839	5.44989
2/3/2019	6/29/2019	7/13/2019		3.45599	4.24979	0.38491	3.70417		4.22641	5.2094	0.40681	4.86017
2/3/2019	7/13/2019	7/27/2019		3.21441	4.14975	0.29137	3.23014		3.6533	4.65981	0.3051	4.22684
2/3/2019	7/27/2019	8/10/2019		3.97429	4.90866	0.40385	4.59309		4.88294	6.15142	0.40695	5.6005
2/3/2019	8/10/2019	8/24/2019		3.60252	4.74421	0.33741	4.1201		3.96818	5.31245	0.301	4.45545
2/3/2019	8/24/2019	9/7/2019		4.21925	5.49902	0.37109	4.47274		4.84816	6.37018	0.40424	5.51008
2/3/2019	9/7/2019	9/21/2019		4.36449	5.97778	0.27712	5.61754		4.97698	6.44312	0.35246	5.80822
2/3/2019	9/21/2019	10/5/2019		4.12437	5.49037	0.28206	4.89339		5.18765	6.675	0.304	5.7266
2/3/2019	10/5/2019	10/19/2019		3.92942	5.17676	0.26816	4.87593		4.60075	6.02774	0.33191	5.37577
2/3/2019	10/19/2019	11/2/2019		4.33851	5.63842	0.37071	4.70461		4.68147	6.13047	0.37808	5.12344
2/3/2019	11/2/2019	11/16/2019		3.78225	4.98868	0.5307	4.27512		3.68205	4.78437	0.44857	3.94353
2/3/2019	11/16/2019	11/30/2019		3.93161	5.21283	0.33388	4.35878		4.82578	6.44155	0.31377	5.87683
2/3/2019	11/30/2019	12/14/2019		4.62146	5.86276	0.33967	5.45258		5.77677	7.26453	0.36048	6.62355
2/3/2019	12/14/2019	12/28/2019		5.26162	6.7696	0.33648	5.69958		5.81207	7.52422	0.35551	6.18918
2/3/2019	12/28/2019	1/11/2020		5.68899	7.37231	0.31249	6.57497		5.73377	7.31147	0.31577	6.71084
2/3/2019	1/11/2020	1/25/2020		4.72063	6.19776	0.3679	5.4888		5.89992	7.5936	0.45047	6.32192
2/3/2019	1/25/2020	2/8/2020		4.68037	6.08175	0.3034	5.58729		6.2476	7.93726	0.34638	7.11459
2/3/2019	2/8/2020	2/22/2020		4.52902	5.67634	0.38353	5.08573		4.88132	6.16737	0.41945	5.21899
2/3/2019	2/22/2020	3/7/2020		4.75405	5.95934	0.27934	5.03072		5.39278	6.65085	0.30835	6.00078
2/3/2019	3/7/2020	3/21/2020		4.44444	5.68259	0.42892	3.87716		4.4461	5.7254	0.37622	4.76332

Figure 15: N-Beats vs. ARIMAX 2-week Error Metrics

Task 2 (UI/ Front End)

The overarching goal of the user interface side of the project was to develop a potential model for Medecipher to implement into their current website to help schedule and adjust hospital nursing schedules. Originally, we were expected to code and develop a website in AngularJS, but after discussing our lack of web development experience with our clients, our project was changed to help them think of design ideas for the current website. We have developed a few different models, all with different layouts and user interface options, and gone through the strengths and limitations of each of them to assist Medecipher in deciding what to fully implement.

Unimplemented Features

Task 1 (Forecasting model):

- Explore combining models to improve error metrics

Task 2 (UI/Frontend):

- We did not end up using AngularJS for our final project, as learning and implementing AngularJS was a little out of our experience area.

Lessons Learned

- Training deep learning models takes a long time. Increasing the number of epochs produces more accurate forecasts, but it can also result in the model taking hours to train;

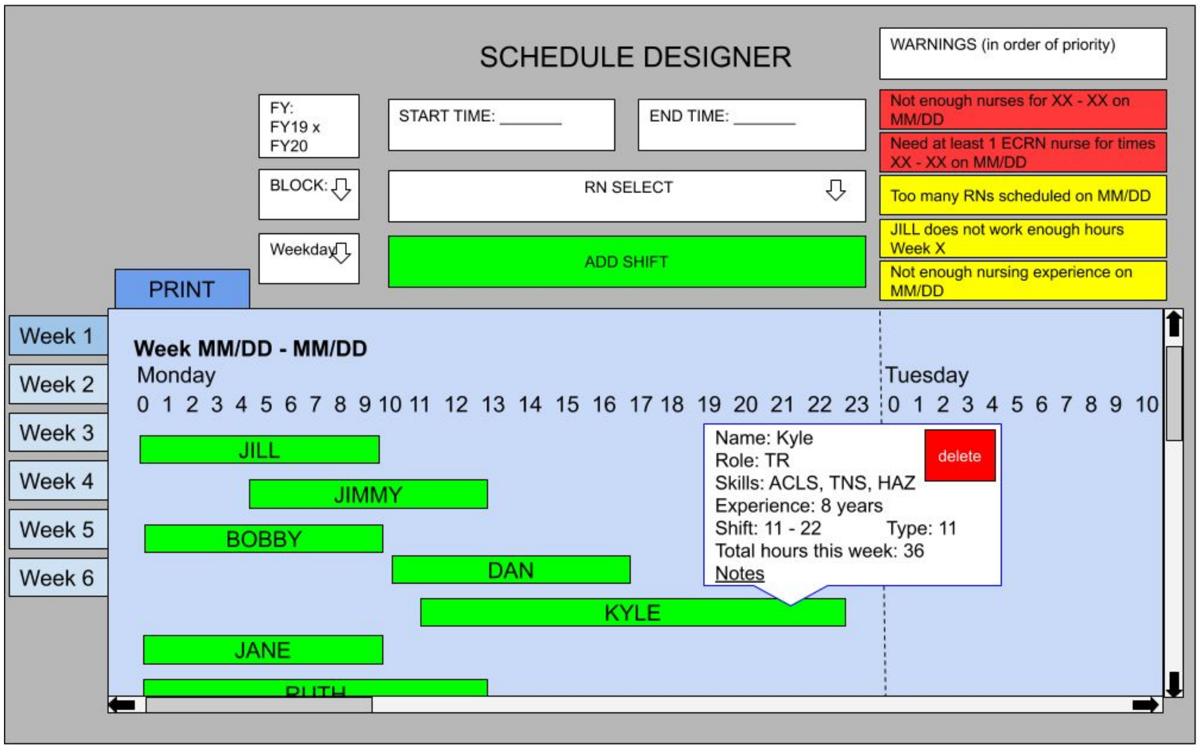
this is also true for several other hyperparameters. In general, we can create pretty accurate forecasts but the tradeoff is increased training time.

- To code in AngularJS, the user must be completely organized in every aspect of their project. It is very difficult to develop a website that is actually worth implementing just because of how many different files are required for one page.
- Between efficiently communicating, understanding clear goals, and delivering a product, working for a real-world client is not easy.
- Staying organized over the course of a project is very helpful in determining what is feasible by the end of the project.
- Communicating with your client is truly the best way to progress in the project. If there are any questions at all, asking our client would immediately clear up issues and problems.
- Make sure your client understands what you are capable of so that they have realistic expectations.
- Python is a very useful language for constructing deep learning models. The programming language offers several libraries and packages with pre-existing deep learning models already built in.

VII. Appendix

All Front End Mockups

Design 1



SCHEDULE DESIGNER

WARNINGS (in order of priority)

- Not enough nurses for XX - XX on MM/DD
- Need at least 1 ECRN nurse for times XX - XX on MM/DD
- Too many RNs scheduled on MM/DD
- JILL does not work enough hours Week X
- Not enough nursing experience on MM/DD

FY: FY19 x FY20

START TIME: _______

END TIME: _______

BLOCK:

RN SELECT

Weekday

ADD SHIFT

PRINT

Week 1
Week 2
Week 3
Week 4
Week 5
Week 6

Week MM/DD - MM/DD

Monday

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

Tuesday

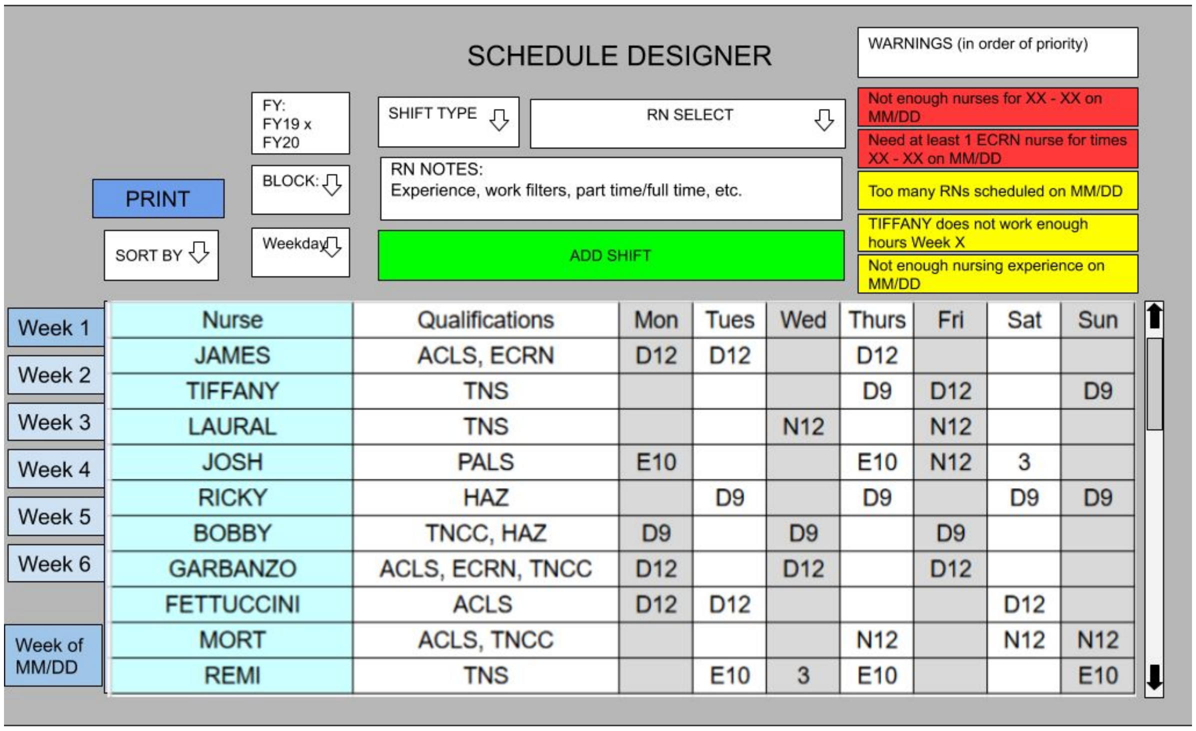
0 1 2 3 4 5 6 7 8 9 10

JILL
JIMMY
BOBBY
DAN
KYLE
JANE
RUTH

Name: Kyle
Role: TR
Skills: ACLS, TNS, HAZ
Experience: 8 years
Shift: 11 - 22 Type: 11
Total hours this week: 36
Notes

delete

Design 2



SCHEDULE DESIGNER

WARNINGS (in order of priority)

- Not enough nurses for XX - XX on MM/DD
- Need at least 1 ECRN nurse for times XX - XX on MM/DD
- Too many RNs scheduled on MM/DD
- TIFFANY does not work enough hours Week X
- Not enough nursing experience on MM/DD

FY: FY19 x FY20

SHIFT TYPE

RN SELECT

PRINT

BLOCK:

RN NOTES:
Experience, work filters, part time/full time, etc.

SORT BY

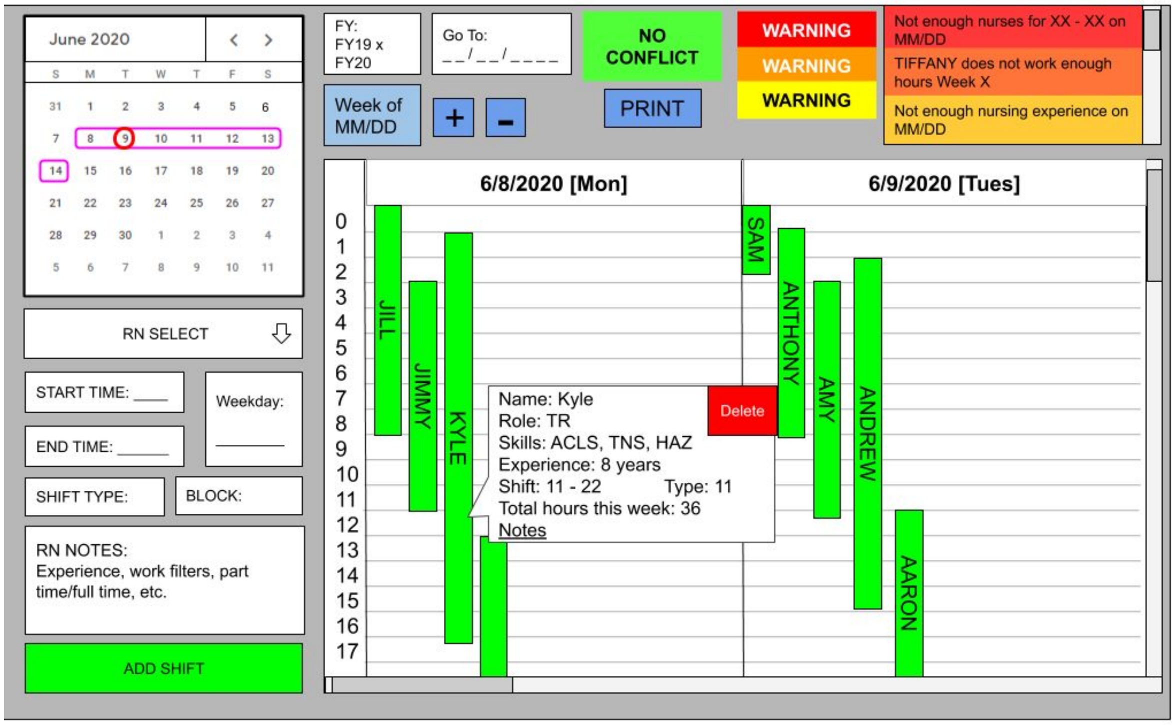
Weekday

ADD SHIFT

Week 1
Week 2
Week 3
Week 4
Week 5
Week 6
Week of MM/DD

Nurse	Qualifications	Mon	Tues	Wed	Thurs	Fri	Sat	Sun
JAMES	ACLS, ECRN	D12	D12		D12			
TIFFANY	TNS				D9	D12		D9
LAURAL	TNS			N12		N12		
JOSH	PALS	E10			E10	N12	3	
RICKY	HAZ		D9		D9		D9	D9
BOBBY	TNCC, HAZ	D9		D9		D9		
GARBANZO	ACLS, ECRN, TNCC	D12		D12		D12		
FETTUCCINI	ACLS	D12	D12				D12	
MORT	ACLS, TNCC				N12		N12	N12
REMI	TNS		E10	3	E10			E10

Design 3

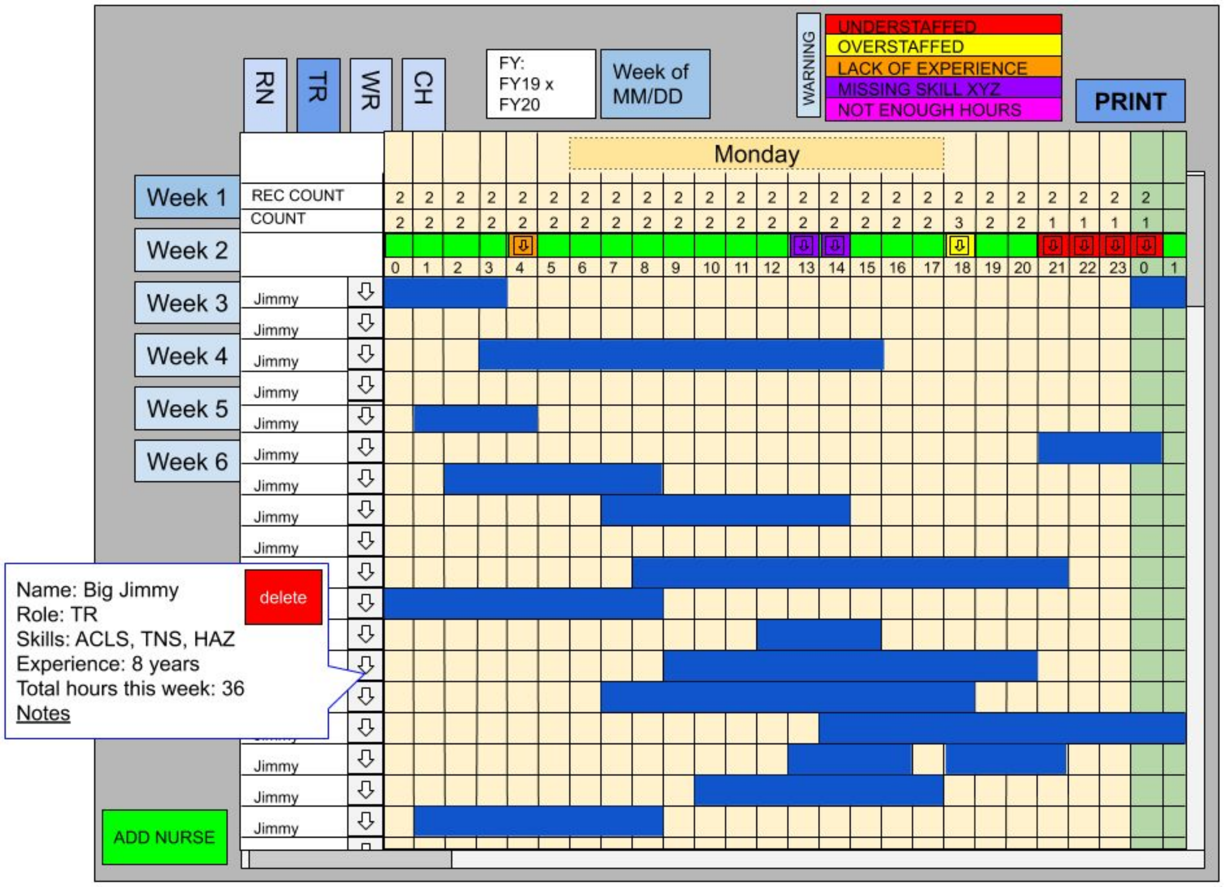


RN NOTES:
Experience, work filters, part time/full time, etc.

Not enough nurses for XX - XX on MM/DD
TIFFANY does not work enough hours Week X
Not enough nursing experience on MM/DD

Name: Kyle
Role: TR
Skills: ACLS, TNS, HAZ
Experience: 8 years
Shift: 11 - 22 Type: 11
Total hours this week: 36
Notes

Design 4



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	0	1
REC COUNT	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	
COUNT	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	3	2	2	1	1	1	1	

Name: Big Jimmy
Role: TR
Skills: ACLS, TNS, HAZ
Experience: 8 years
Total hours this week: 36
Notes

Error Metrics

Mean Absolute Error:

$$\mathrm{MAE} = \frac{1}{n}\sum_{h=1}^{n} |e_{T+h}|$$

Root Mean Squared Error:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{h=1}^{n} (e_{T+h})^2}.$$

Mean Absolute Percent Error:

$$\mathrm{MAPE} = \frac{1}{n}\sum_{h=1}^{n} |p_{T+h}|,$$

Mean One-Sided Error:

max(Actual Count - Forecast, 0) / (number of positive values)