Analyzing and Forecasting U.S. Immigration Trends

Group 7: Alan Lin , Trevor Petrin , Ganesh Kumar Ramar , Mingyu Chen

Repository: alanklin/AA-Capstone: Boston College Applied Analytics Capstone Project

This week, we aimed to improve our cross-validation method before moving forward with our model building process. As we mentioned last week, we hadn't properly implemented the Hyndman cross-validation method through windowing so we noticed a significant drop in performance in the latter half of the test set. We've now implemented a more effective validation set and have noticed some improvements in our current metric of performance (MAPE).

Tying back to last week, we showed how the Neural Network models performed marginally better than the ARIMA, Transformer, and LSTMs models on the basis of having a lower Mean Absolute Percentage Error at each time step in the testing set. We continue building on this model through hyperparameter tuning and also built a custom LSTM model for the sake of providing more model options. For the purposes of this week, let's stick with the Neural Network and the different iterations we've done with hyperparameter tuning. We did a total of 27 different configurations using these three hyperparameters.

Hyperparameters Tuned		
Learning Rate (α)	Window Size	Dropout Rate
• 0.001 • 0.0005 • 0.0001	• 12 • 18 • 24	• 0.3 • 0.4 • 0.5

To perform cross validation, data windows were used to create input ready data sections for the model to be trained upon with the immediately following 12 data points used as the dependent variable for that section. A function was created to window the data with a certain number of training nodes followed by 12 output values which the model will learn to pick. The number of independent observations was used as the main hyperparameter to tune using the values of 12, 18, and 24 months to predict the next 12. This impacted the structure of the Neural Network input, as the number of input nudes must match the size of the number of independent observations. This splitting technique allowed for each individual model to be more thoroughly trained upon what data was available until the end of the train data.

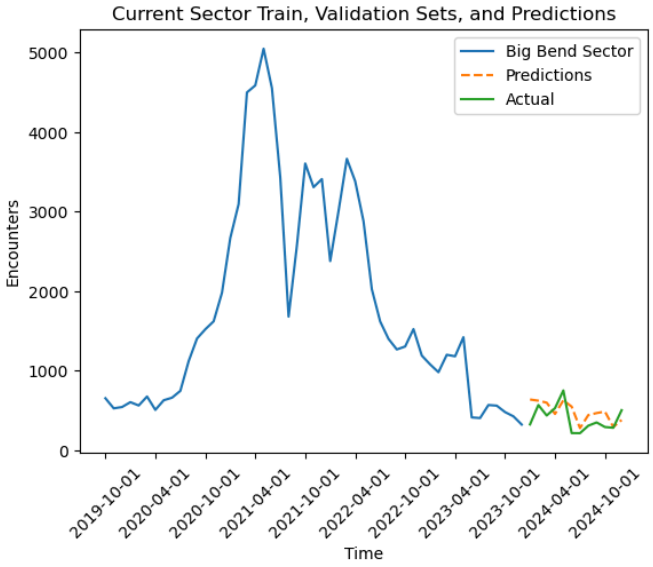
There will also need to be 41 separate models trained using these techniques, one for each individual sector, as the trends and seasonality on the northern border are not likely to match those on the southern border. The tuning process will allow for there to be multiple models trained upon each sector with the best model being chosen to forecast the sector. There will be

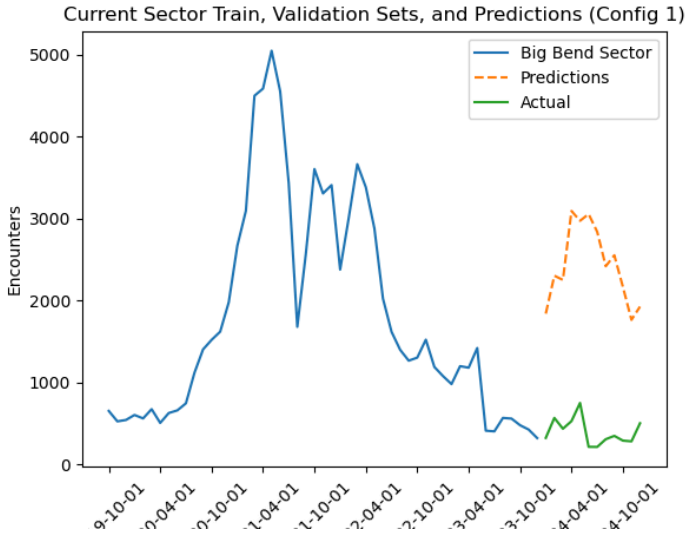
multiple hyperparameters to tune with each model as well including the learning rate, the window size, and the dropout rate per layer. Other factors such as the optimizer and the number of nodes were not tinkered with.

As for the model implementation details, TensorFlow was used to create the back end of the model, having an input layer, 3 hidden layers, and 12 output nodes, preparing the model for 12 points of output. The hidden layer sizes, in order, were 512, 256, and 128, to allow for the model to better pick up on the more sophisticated trends within the data as opposed to a simpler model. Several nested for loops were used to iterate over the data to create models, using the output MAPE to determine the optimal performance. This metric was used to better keep the error in perspective sector-by-sector because an error of 1000 individuals may be 10% or 100% off of the mark depending upon the sector, contextualizing better than a RMSE measurement.

The training process for the neural network model in the Big Bend Sector example, using a randomly configured model, exhibited inconsistent performance—likely due to an overly high learning rate causing fluctuations. However, the validation results showed steady improvement throughout the process. We would expect a similar performance with that learning rate across sectors.

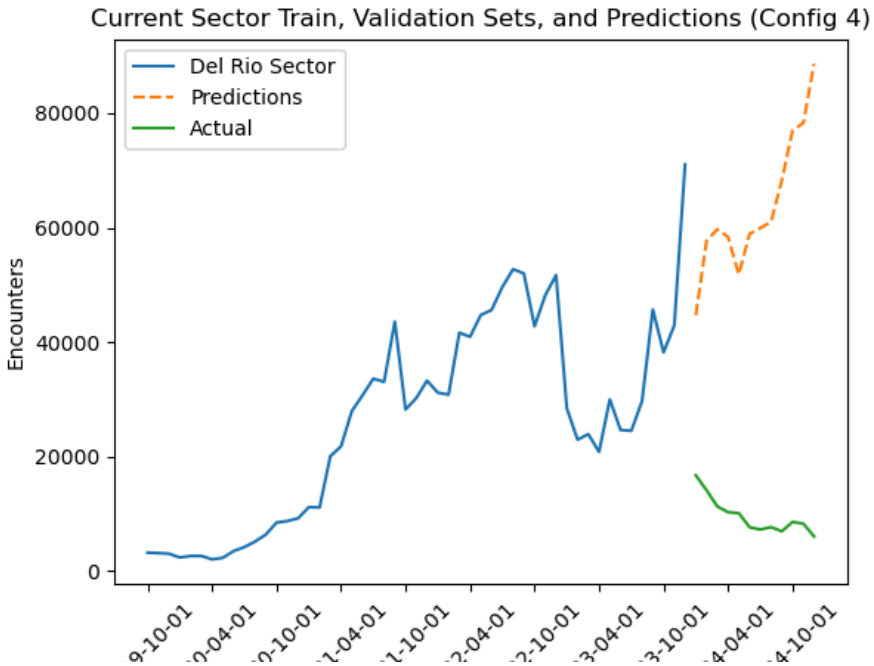
For selecting the most effective model for each sector, the average MAPE across all observations will be used to compare the performance. It was remarkable, the difference that the model's hyperparameters had upon the performance.



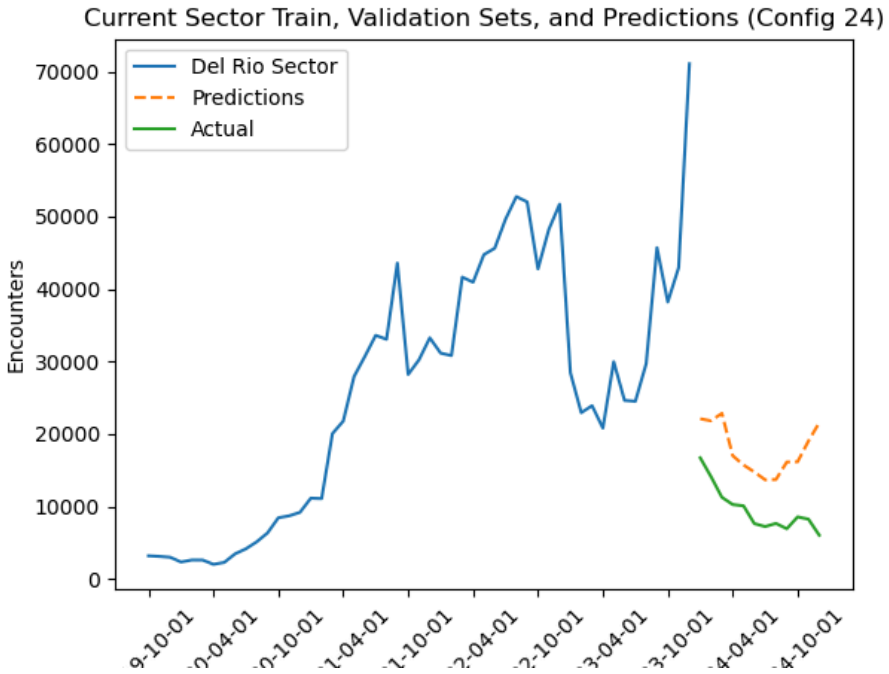


As can be seen with the Big Bend Sector predictions with a window of 24 and hyperparameter set of {0.001, 0.3}, the test predictions mirror the actual performance extraordinarily well for picking up the trend. However, when the model's parameters are changed such that the window size is down to 12 months, the performance decreases dramatically. However, this is not consistent with every model where there are combinations that work the opposite. Thus each individual model is tuned out of conjunction with the others.

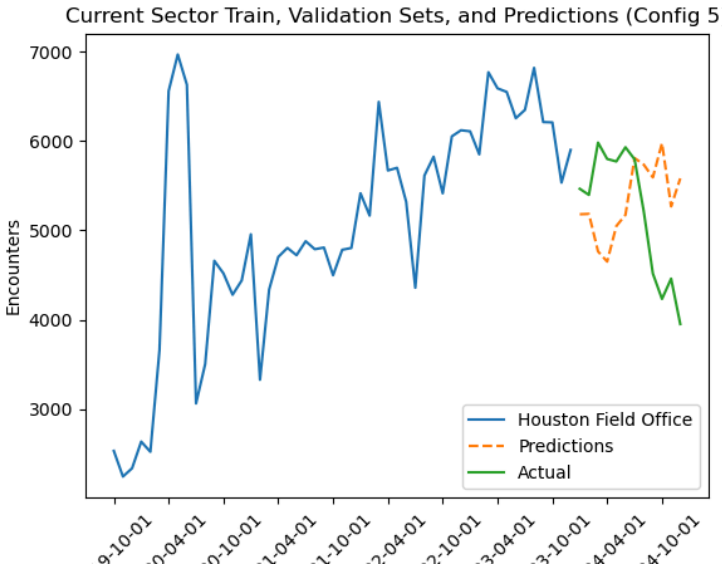
For the overall trend evaluation, once we were able to select our best models using the hyperparameter tuning, further evaluation was done upon which models needed more work because of difficult data. As seen below, the Del Rio sector was pictured at a terrible time to forecast with the spike in immigration in December to nearly double what the previous observation was 4 months prior, followed by a 3-year minimum. While some were wildly off, other models did come close when the window size was increased to 24. Still, it seems that each sector might require different hyperparameters as each combination of configurations had differing results.

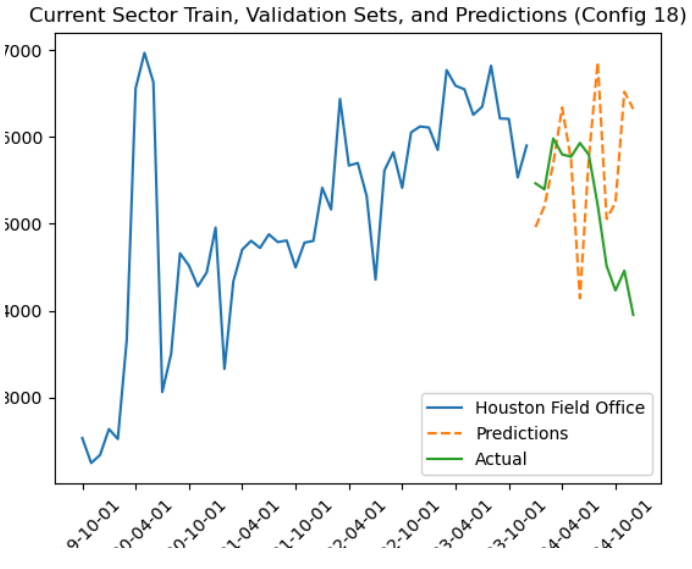


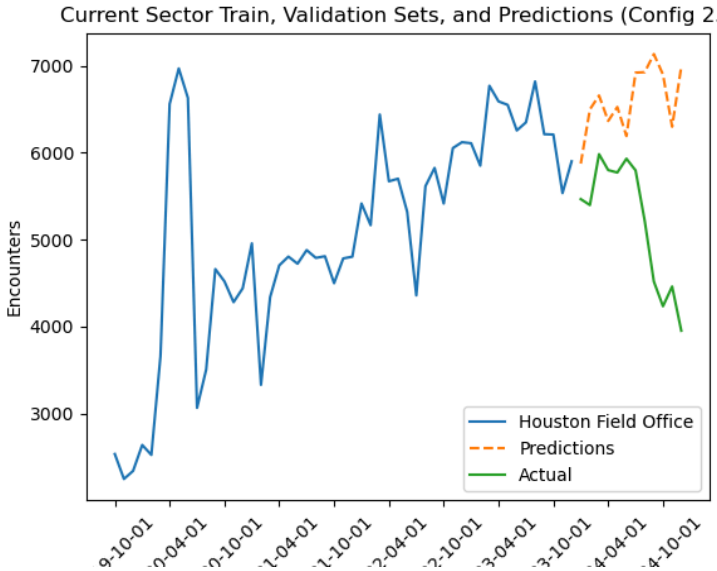
Window size of 12 - led the model to predict a continuation of the upward trend.



Window size of 24 - was much closer to the actual values!







There were other sectors where every combination struggled and neural networks may be too complex for this dataset. As is pictured above with the Houston Field Office, most, if not all, models were unable to correctly pick up on the trend of the model. In this case, a more basic model such as ARIMA or ETS may capture the recent trends better than the Neural Network.

How this issue may be addressed in future iterations of the model building process would be to introduce lag in between the train and validation data such that the spike is not accounted for

while making the predictions upon the test data. This aspect may be tuned as a future hyperparameter, with the methodology behind this being that the model avoids the most recent spike as the model input such that it is not reflected in the model's trend. This can be further experimented with next week when the ARIMA models are built.

As for the axis, we are fully aware that the labels were cut off when we saved to png, this will be fixed next iteration. It should be a simple fix by using plt.tight_layout(). However, it would have meant regenerating the images, which took over 4 hours to complete with 27 different configurations, so we unfortunately ran out of time this week.

We initially saved the models themselves, however, in hindsight, a more quantitative solution would have been to store the results as a flat file as opposed to a graph to perform further analysis upon the model's completion. Unfortunately for this week, though, it meant that we did not necessarily have room to incorporate the numeric results more into separate graphs or other ways to display the data. In hindsight, and in the weeks going forward, this code will be altered to save the data using .pkl files for their efficiency with space and so we do not go over the github space limit. Graphing will be reserved for anomalies and examples as opposed to a uniform graph for all models at all data points. We'll aim to create graphs that include multiple lines on one graph as opposed to having several spread out across future reports.

Upon evaluation of these results, the team considered the possibility this week that the length of the forecast may be too long to effectively make a good prediction about what will happen with reasonable accuracy. This was determined using the MAPE average error by time point graph as well as the average RMSE graph, there is a clear spike in error on the back 4 months of the forecast. For funding purposes with our use case, 6 months is still a reasonable outlook, and further discussions will need to be had about focusing on a solid 6 month forecast or whether to keep training on the current 12 month test set. With how the data is structured, it does not help that the train/test split point at December 2023 and January 2024 is where there is a significant drop, changing the trend immensely.

We don't want to move the goalpost, but the shorter-term forecasting approach would likely provide more value to the project stakeholders, giving better reason to to trust the forecasts that we are delivering. We are still debating where to put the test split and whether to keep it at the current position to give a difficult dataset but focus on making the results land well or moving the split to have a more consistent trend and likely better forecast. In other words, it may be more valuable to provide less of a forecast that gives better information. So far, we've gotten a thumbs up from Doc. Arvind about decreasing the test set to 6 months instead of the full 2024 calendar year and we'll see how these results turn out in the next few weeks.