Forecasting Street and Sidewalk Cleaning Services in San Francisco

By Jared Schober (50% of all work) and Peter Amerkhanian (50% of all work)

Context and motivation

In 2007, the city and county of San Francisco established SF311, an easily accessible and memorable phone number for accessing public services. Since July 2008, SF311 has collected and published data on 311 calls and requests. There are many different types of requests included in the data, such as maintenance (e.g. downed tree, broken streetlights), noise complaints, and general inquiries. The most common service request type is street and sidewalk cleaning (SSC) requests, with more than 2 million observations recorded in SF311's publicly available dataset. When an SSC request is received, the 311 hotline dispatches a Public Works crew to clean the street and/or sidewalk at the requested location.

In this analysis, we produce accurate forecasts of the number of SSC requests received via SF311 on an hourly basis. Our results can be applied to improve the provision of SSC services by keeping staffing levels responsive to call demand. With accurate forecasts, SF311 can adequately staff its call center and Public Works' cleaning crews, avoiding having too many or too few personnel to handle call volume.

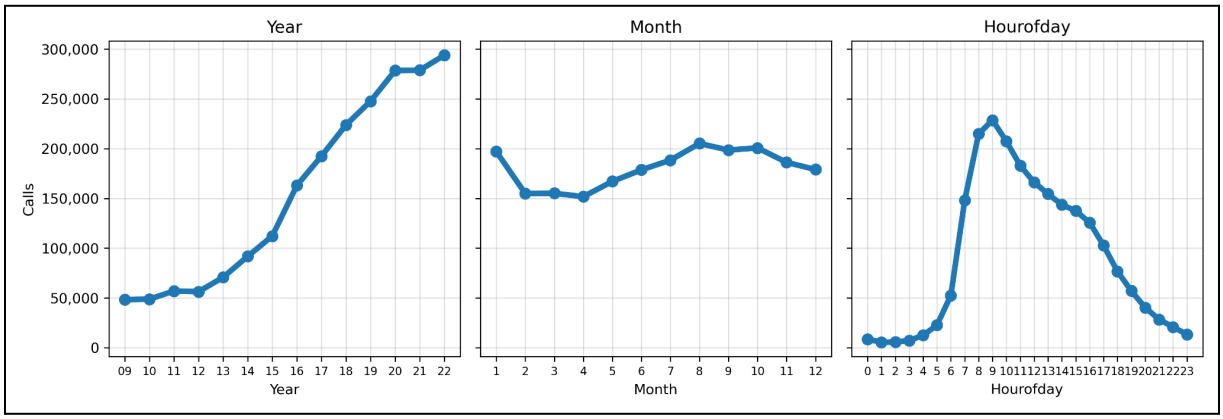
Data

Figure 1: Descriptive statistics for call counts at the hour-level

N	Std. Dev.	Mean	Min	25th %	50th %	75th %	Max
122,712	22.592	17.627	0	2	8	25	553

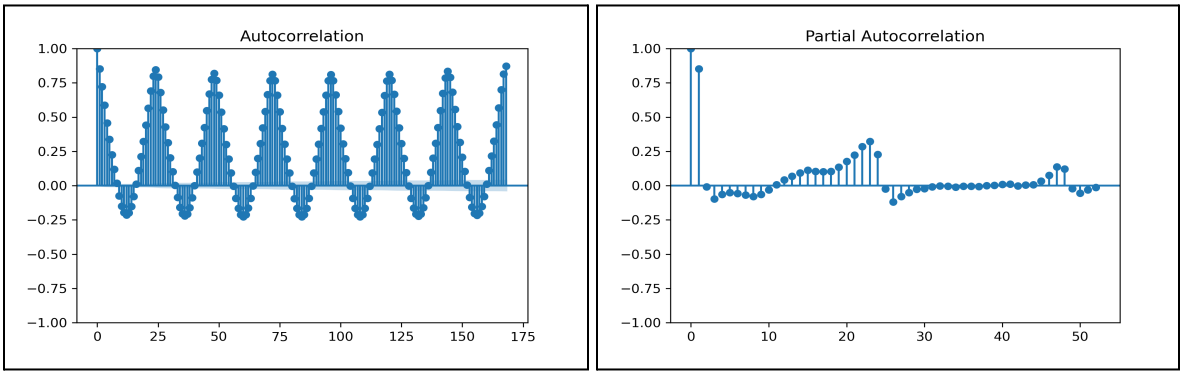
We examine data from 1/1/2009 to 12/31/2022, aggregated at the hour-level. The number of calls is variable across hours, and we find that much of this variation is driven by strong seasonal patterns and a secular increase in calls across years. The use of 311 for SSC requests grew considerably over this time period – from 50K to 300K calls annually. We anticipate that this growth will complicate forecasting, as it is unlikely that a model will be able to anticipate the sharp growth periods followed and/or preceded by plateaus. In our modeling approach, we focus on the pronounced hourly trend.

Figure 2: Seasonal Trends: Annual, Monthly, Hourly



We further explore seasonalities by computing autocorrelation and partial autocorrelation across hour-level time lags. There is strong, statistically significant autocorrelation at each 24 hour increment that persists well past the one week displayed in Figure 3 below. However, partial autocorrelation diminishes in significance after about 48 hours. From these processes, we determine that our models will make use of the previous 48 hours of call data.

Figure 3: Autocorrelations



Data processing and training

To process the 311 call data, we standardize the data on a yearly basis. We also clip outliers in excess of three standard deviations of the mean. This is significant because, although there are very large outliers, they are rare and would be difficult for the models to interpret. We construct a time-series-appropriate cross validation strategy, implementing train-test splits of the data on a yearly basis, training each model on the current year (e.g. 2009), and then forecasting on the next year's data (e.g. 2010). Standardization is fit using the same train-test splits.

Given that we are predicting a count, we evaluate our models using the root mean squared error (RMSE), the R^2 score, and the maximum residuals. RMSE expresses a model's average error, R^2 measures errors at an aggregate level, and the maximum residual gives us a sense of the "worst-case" error. Each of these metrics are averaged across the year-folds in our cross validation strategy.

Modeling Approaches

We devise two modeling approaches. The first is generating long-run predictions, in which we seek to evaluate a model's ability to predict one year of calls. The second is next-hour prediction, in which a model is given the previous 48 hours and asked to predict the next hour.

Long-Run Predictions

In this case, each model learns from one year's worth of data, then is tasked with predicting the entire next year of data, year-by-year. This is not a practically realistic forecasting task – if a model were in use by Public Works, it would be updated periodically with new data throughout the year – but it gives us an idea of how far each model can make predictions before quality degrades. For this task, we select the following models:

- Autoregressive Integrated Moving Average (ARIMA) with order (48, 1, 0)
- OLS regression trained on 48 hours of lags
- A Naive Baseline that predicts the value from 48 hours ago
- A Random Forest Regression trained on 48 hours of lags

ARIMA is chosen because it is arguably the most common method for time series forecasting. The selection of other models represents a mixture of approaches to learning from the lag data. The random forest will learn nonlinear patterns between the lags and the outcome and will develop complex webs of interactions, whereas OLS will be restricted to learning linear relationships. Given that the data contain strong linear relationships, we believe a linear model will be strong in this task. None of these models have access to information about the next year, and their predictions are made using a chaining structure. Each test prediction is generated sequentially by having the fit model make a prediction based on the following:

1. Use the last 48 hours of the training data
2. Use the last 47 hours of the training data and the first predicted hour of the test
3. Use the last 46 hours of the training data and the first two predicted hours of the test, and so on.

ARIMA builds a variation of this process into its own algorithm, while we program our own implementation of this chaining for all other models.

Next-Hour Predictions

Producing next-hour predictions is the policy-relevant modeling goal in this paper. We would seek to operationalize this model at Public Works' dispatch center, so that they know where to send cleaners in the next hour. For next-hour predictions, our forecasting is set up as a more traditional machine learning problem, in which we are given a row from a design matrix of features, in this case lag data containing the previous 48 hours of trend, and the model outputs a scalar prediction, in this case the next hour's call volume. We select a mixture of linear and nonlinear ensemble methods to provide a spectrum of results.

- A Naive Baseline that predicts the number of calls at this time 48 hours ago
- Linear regression to provide another baseline
- Lasso regression to explore to what degree regularization helps performance

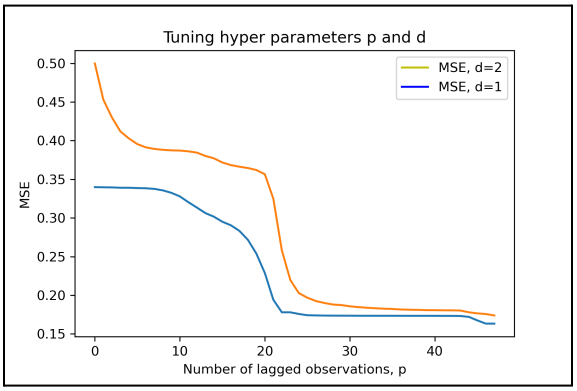
- Random forest regression to explore how nonlinearity, interactions, and general complexity affect performance
- A Recurrent Neural Network (RNN) to examine how using a highly complex deep learning approach affects performance

We additionally considered using the Long Short Term Memory architecture for our RNN, but found that we likely were using too little training data (~8,000 observations per training fold) and focusing on too little of a prediction window (one observation) to take full advantage of the architecture. We also considered ARIMA for this next-hour prediction task, but each run of the ARIMA model is computationally intense and it would have been impossible to evaluate in a timely manner.

Hyperparameter tuning / model improvements

Lasso Regression:
For lasso regression, we tuned the alpha hyperparameter to evaluate how much regularization should be included to account for overfitting in the OLS model. We tested 10 values of alpha from 0.0001 to 10, and selected the value that minimized RMSE. We found that a value of 0.0001 for alpha was optimal on each test set, indicating that OLS was likely picking up on true linear trends in the data.

ARIMA:
For the ARIMA model, we utilize the autoregressive and integrative parts of the model, and do not use the moving average part. Therefore, we tuned two of the three ARIMA hyperparameters: p, the number of lag observations to include in the model, and d, the degree of differencing. P represents the number of hourly lags to include, and d represents the number of times observations should be differenced to transform non-stationary data into stationary data. We do not include q, the moving average window size, in our model. We found that RMSE increased precipitously until 24 hours, and then plateaued before slightly dipping at 48 hours.



We also found a degree of differencing of 1 to perform better than a value of 2. These results are expected from the autocorrelation and partial autocorrelation charts above. For our ARIMA model predictions, we used the tuned optimal order of (48, 1, 0)

Random Forest Regression:
For the random forest regression, we focus on tuning the maximum depth of each decision tree in the ensemble and the number of estimators used in the random forest. We chose these two hyperparameters because they have direct effects on the complexity of individual trees in the forest, and the size of the forest, two key indicators of overall ensemble complexity. Given more

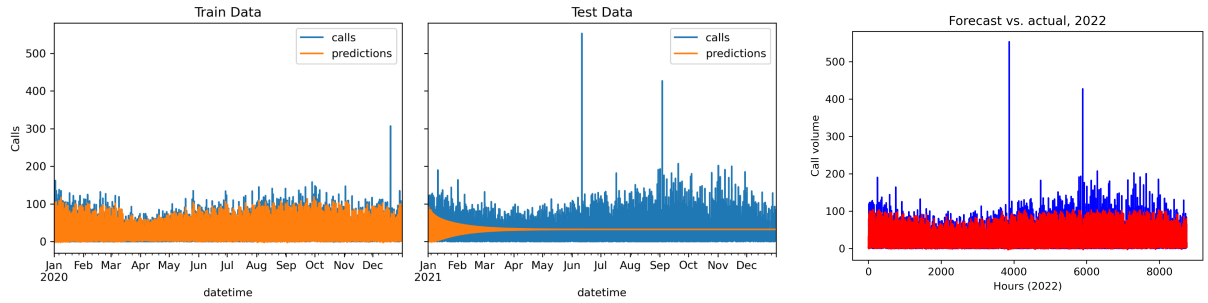
time, we would focus on tuning more hyperparameters for the random forest, such as the minimum number of samples required to split an internal node or different split criterion.

Recurrent Neural Network (RNN):
For the RNN, we investigated many different hyperparameters. We experimented with the number of layers or depth of the network, number of nodes per layer, adding dropout layers between RNN layers, the activation function, optimizer, and loss value. There are additional factors, such as regularization, as well. However, even training on one year's worth of data took upwards of an hour per run, so it was difficult to systematically tune the hyperparameters. The model we report below utilizes four SimpleRNN layers of 200, 150, 100, and 50 nodes, with 0.50 dropout layers following. We use "relu" activation, the adam optimizer algorithm and mean squared error loss. To fully optimize the RNN, we would continue to vary each of these hyperparameters, including adjusting the percentage dropout, and deepening the network.

Results

Long-run predictions:
We find that the ARIMA model is uniquely well-suited to longer term predictions. ARIMA's strengths make intuitive sense – while all of the models had access to 48 hours lags, ARIMA is the only model with "integration" – it differenced the data to produce stationary trends. With stationary data, the ARIMA model was able to use the 48 hour lags to much more accurately forecast long term trends, capturing an average of 56% of the variation in true test-set variation across folds. With non-stationary data, all other models degraded considerably over time. The last fold is pictured in Figure 4.

Figure 4: A. OLS Train and Test Performance **B. ARIMA Test Performance**



We find that R^2 is the metric that best captures the degradation in prediction quality for all models besides ARIMA. The negative R^2 values indicate that a horizontal line can fit the data better than models other than ARIMA.

Model	Avg RMSE	Avg R^2	Avg max residual
ARIMA	11.37	0.56	161.376
OLS	17.46	-0.041	166.23
Random Forest	20.277	-0.394	170.54
Naive Baseline	22.97	-0.75	179.33

Next-Hour Predictions:
We find that the random forest regression performed best out of our next-hour models. We believe that this is because the random forest doesn't restrict itself to linear trends, like OLS and

Lasso, and is able to learn the non-linearities in the data. The random forest model had the lowest RMSE, R^2 and average max residual values compared to the other models. Linear regression and Lasso were both able to outperform the naive baseline of predicting the call volume of the current hour yesterday. However, the RNN significantly underperformed the naive baseline, with significantly higher RMSE and average max residual, and significantly lower R^2. We believe that with additional tuning, the RNN model would be more competitive than the regression models, and potentially more competitive than the random forest. However, it takes significantly longer to run and is far more difficult to tune due to computational limitations.

Figure 5: Next-Hour model results

Model	Avg RMSE	Avg R^2	Avg max residual
Baseline	12.58	0.379	171
Linear Regression	8.29	0.595	157.33
Lasso Regression (*α=0.001*)	8.28	0.594	157.30
Random Forest (*max_depth=None, estimators=500*)	7.95	0.606	153.60
RNN	62.13	0.18	398.465

Applying modeling to Zip-Code Level Data

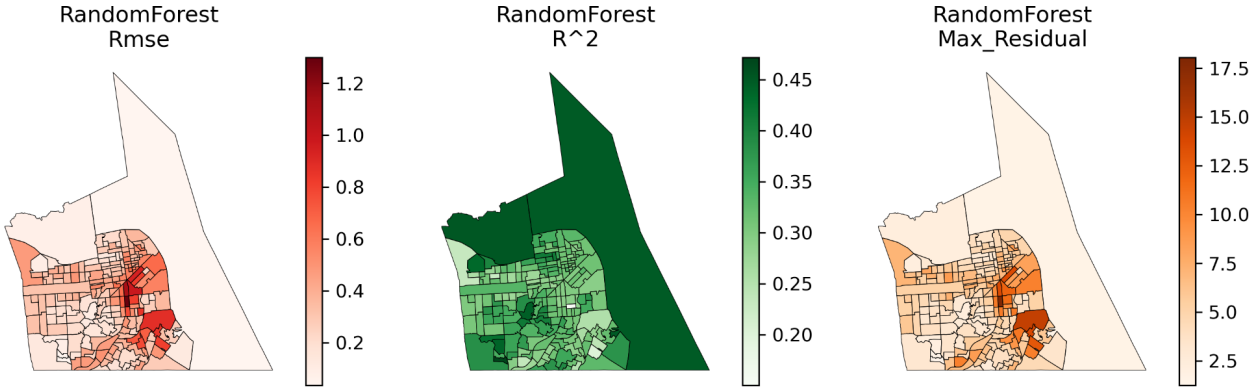
To operationalize one of our machine learning models for 311 service dispatching, we proceed to incorporate the variation in calls across San Francisco neighborhoods. We evaluate the performance of OLS and the random forest (RF) regression at the census tract level, training and cross validating the performance of each model for all 196 census tracts in the city. Metrics are recorded in Figure 5 below.

Figure 6: Census tract model results

	RF RMSE	RF R^2	RF Max Residual	OLS RMSE	OLS R^2	OLS Max Residual
mean	0.368	0.324	5.659	0.357	0.364	5.630
std	0.201	0.045	2.619	0.198	0.028	2.593
min	0.035	0.150	1.249	0.0347	0.296	1.250
25%	0.230	0.297	3.947	0.223	0.346	3.969
50%	0.327	0.317	5.017	0.312	0.362	4.998
75%	0.456	0.345	6.512	0.445	0.379	6.508
max	1.300	0.472	18.045	1.347	0.478	18.27

Training at the census tract level gives each model access to much less variation in calls to learn, and thus the performance of OLS, a parametric model that can perform well with small data, is much closer to our previous best model – the random forest, which is less adept with small data. Indeed, in terms of R^2, OLS is a better model in this low-information setting. We examine the spatial distribution of metrics for the random forest regression in Figure 6. Generally speaking, the model performs best in sparsely populated regions like the presidio and treasure island and is weaker in denser areas like downtown. However, the RMSE and max residuals can be misleading in this context, as areas like downtown have significantly more calls than say the presidio, and will necessarily have higher RMSE and maximum residuals.

Figure 7: Spatial variance in model results



Limitations and Conclusion

We found that almost all of the machine learning models we applied to this time series forecasting task outperformed reasonable naive baselines by a meaningful margin. For long-term forecasting, ARIMA significantly outperformed both the naive baseline and all other models. For short-term forecasting, random forest regression performed the best.

It is important to highlight two important limitations of applying machine learning models to time series forecasting. First, models don't handle outliers and turning points well. This is a fundamental problem, as the model's errors will be greatest when calls are unexpectedly spiking. However, it is likely that SF311's call center would have to scramble no matter what when a large call spike occurs. For our goal of accurately forecasting for staffing and cost considerations, missing outliers is less concerning. Second, from our summary statistics, it is clear that there is moderate seasonality in the data. This is most pronounced on a daily basis, but also occurs across months and years as well. Since our models look at either just the last 48 hours or the last year of observations, the models are not able to fully account for this seasonality.

In conclusion, we found that utilizing machine learning methodologies to forecast 311 Street and Service Cleaning calls yields an improvement of 50% for long-term forecasting (ARIMA RMSE of 11.37 compared to naive baseline of 22.97), and 37% for short-term forecasting (random forest regression RMSE of 7.95 compared to naive baseline RMSE of 12.58). SF311 should utilize these methods to improve staffing, reduce costs and better deliver this important service to the public.