Forecasting Street and Sidewalk Cleaning Services in San Francisco

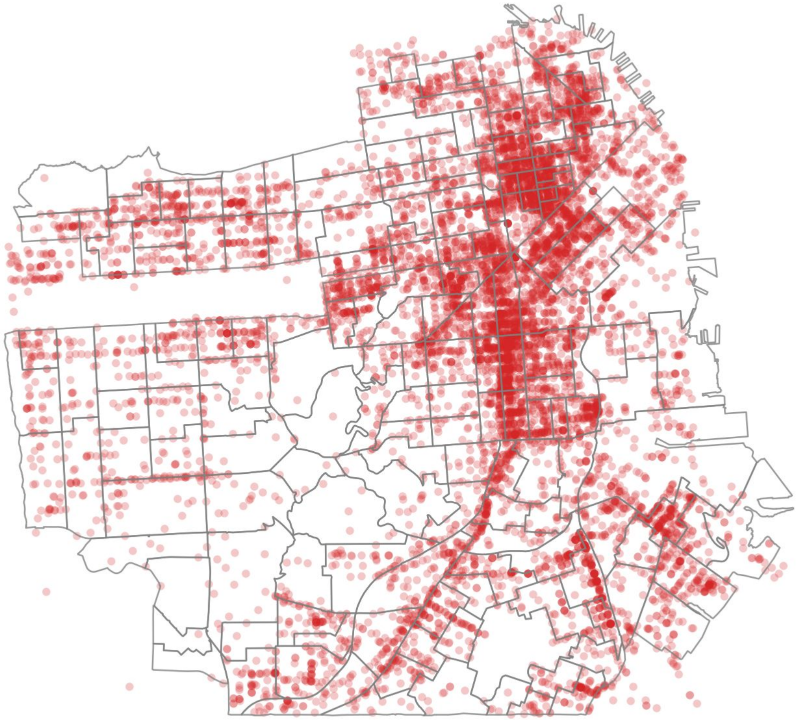
Jared Schober and Peter Amerkhanian

San Francisco 311

“Street and sidewalk cleaning requests are generated internally and **through calls received by the City's 311 customer service center.** [...] Public Works' Radio Room **triages the request to the appropriate crew**.” (source)

We examine call counts **at the hour level** between 1/1/2009 and 12/31/2022:

```
----- Descriptive Statistics -----
count    122712.000000
mean         17.626622
std          22.591759
min           0.000000
25%           2.000000
50%           8.000000
75%          25.000000
max         553.000000
Name: calls, dtype: float64
Sum: 2162998
```



Goals and Applications

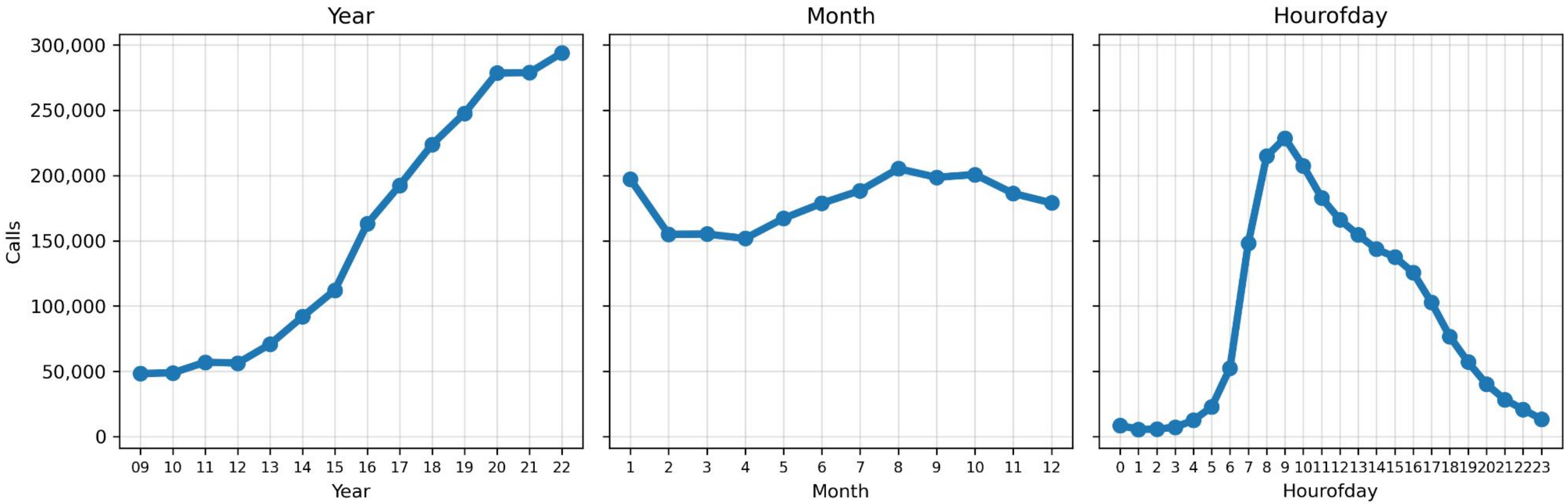
Our goal: Use machine learning to predict the number of requests that the city will receive in any given hour.

Policy applications: Understand how much staffing will be needed for the 311 call center and public works street cleaning teams at a given time, anticipate sidewalk and street cleaning costs.

SF Public Works Clean Corridors

Approach and Challenges

- **311 grew** considerably during this time period.
- We were concerned it would be difficult to forecast within year trends with such a strong secular trend across years.
- We focused on capturing hour-level patterns in our modeling



Data and Cross Validation Approach

Lagged Dataset (48 hours picked via exploration)

	calls	lag_1	lag_2	lag_3	lag_4	lag_5	lag_6	lag_7	lag_8	lag_9	...	lag_39	lag_40	lag_41	lag_42	lag_43	lag_44	lag_45	lag_46	lag_47	lag_48
datetime																					
2009-01-03 00:00:00	0	6.0	0.0	1.0	1.0	3.0	5.0	19.0	20.0	19.0	...	6.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2009-01-03 01:00:00	0	0.0	6.0	0.0	1.0	1.0	3.0	5.0	19.0	20.0	...	4.0	6.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2009-01-03 02:00:00	1	0.0	0.0	6.0	0.0	1.0	1.0	3.0	5.0	19.0	...	10.0	4.0	6.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
2009-01-03 03:00:00	3	1.0	0.0	0.0	6.0	0.0	1.0	1.0	3.0	5.0	...	8.0	10.0	4.0	6.0	1.0	0.0	0.0	0.0	0.0	0.0
2009-01-03 04:00:00	0	3.0	1.0	0.0	0.0	6.0	0.0	1.0	1.0	3.0	...	5.0	8.0	10.0	4.0	6.0	1.0	0.0	0.0	0.0	0.0

Cross Validation: Train, Test, **standardize, clip at ±3std, compute metrics**

	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022
Fold Size	8712	8760	8760	8784	8760	8760	8760	8784	8760	8760	8760	8784	8760	8760

Data and Cross Validation Approach

Lagged Dataset (48 hours picked via exploration)

	calls	lag_1	lag_2	lag_3	lag_4	lag_5	lag_6	lag_7	lag_8	lag_9	...	lag_39	lag_40	lag_41	lag_42	lag_43	lag_44	lag_45	lag_46	lag_47	lag_48
datetime																					
2009-01-03 00:00:00	0	6.0	0.0	1.0	1.0	3.0	5.0	19.0	20.0	19.0	...	6.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2009-01-03 01:00:00	0	0.0	6.0	0.0	1.0	1.0	3.0	5.0	19.0	20.0	...	4.0	6.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2009-01-03 02:00:00	1	0.0	0.0	6.0	0.0	1.0	1.0	3.0	5.0	19.0	...	10.0	4.0	6.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
2009-01-03 03:00:00	3	1.0	0.0	0.0	6.0	0.0	1.0	1.0	3.0	5.0	...	8.0	10.0	4.0	6.0	1.0	0.0	0.0	0.0	0.0	0.0
2009-01-03 04:00:00	0	3.0	1.0	0.0	0.0	6.0	0.0	1.0	1.0	3.0	...	5.0	8.0	10.0	4.0	6.0	1.0	0.0	0.0	0.0	0.0

Cross Validation: Train, Test, **standardize, clip at ±3std, compute metrics**

	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022
Fold Size	8712	8760	8760	8784	8760	8760	8760	8784	8760	8760	8760	8784	8760	8760

Data and Cross Validation Approach

Lagged Dataset (48 hours picked via exploration)

datetime	calls	lag_1	lag_2	lag_3	lag_4	lag_5	lag_6	lag_7	lag_8	lag_9	...	lag_39	lag_40	lag_41	lag_42	lag_43	lag_44	lag_45	lag_46	lag_47	lag_48
2009-01-03 00:00:00	0	6.0	0.0	1.0	1.0	3.0	5.0	19.0	20.0	19.0	...	6.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2009-01-03 01:00:00	0	0.0	6.0	0.0	1.0	1.0	3.0	5.0	19.0	20.0	...	4.0	6.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2009-01-03 02:00:00	1	0.0	0.0	6.0	0.0	1.0	1.0	3.0	5.0	19.0	...	10.0	4.0	6.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
2009-01-03 03:00:00	3	1.0	0.0	0.0	6.0	0.0	1.0	1.0	3.0	5.0	...	8.0	10.0	4.0	6.0	1.0	0.0	0.0	0.0	0.0	0.0
2009-01-03 04:00:00	0	3.0	1.0	0.0	0.0	6.0	0.0	1.0	1.0	3.0	...	5.0	8.0	10.0	4.0	6.0	1.0	0.0	0.0	0.0	0.0

Cross Validation: Train, Test, **standardize, clip at ±3std, compute metrics**

	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022
Fold Size	8712	8760	8760	8784	8760	8760	8760	8784	8760	8760	8760	8784	8760	8760

Data and Cross Validation Approach

Lagged Dataset (48 hours picked via exploration)

	calls	lag_1	lag_2	lag_3	lag_4	lag_5	lag_6	lag_7	lag_8	lag_9	...	lag_39	lag_40	lag_41	lag_42	lag_43	lag_44	lag_45	lag_46	lag_47	lag_48
datetime																					
2009-01-03 00:00:00	0	6.0	0.0	1.0	1.0	3.0	5.0	19.0	20.0	19.0	...	6.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2009-01-03 01:00:00	0	0.0	6.0	0.0	1.0	1.0	3.0	5.0	19.0	20.0	...	4.0	6.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2009-01-03 02:00:00	1	0.0	0.0	6.0	0.0	1.0	1.0	3.0	5.0	19.0	...	10.0	4.0	6.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
2009-01-03 03:00:00	3	1.0	0.0	0.0	6.0	0.0	1.0	1.0	3.0	5.0	...	8.0	10.0	4.0	6.0	1.0	0.0	0.0	0.0	0.0	0.0
2009-01-03 04:00:00	0	3.0	1.0	0.0	0.0	6.0	0.0	1.0	1.0	3.0	...	5.0	8.0	10.0	4.0	6.0	1.0	0.0	0.0	0.0	0.0

Cross Validation: Train, Test, **standardize, clip at ±3std, compute metrics**

etc…

	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022
Fold Size	8712	8760	8760	8784	8760	8760	8760	8784	8760	8760	8760	8784	8760	8760

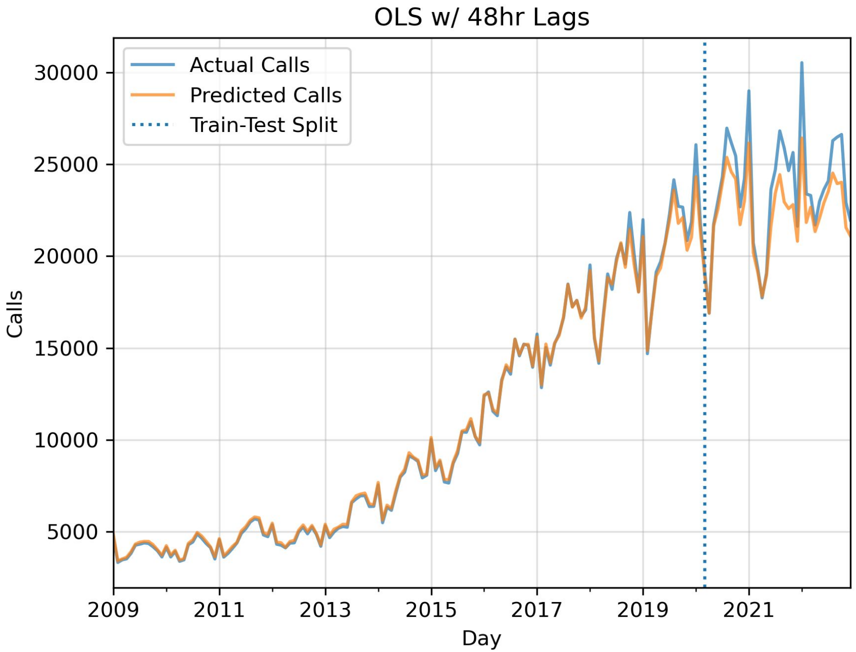
Summary

Model	Avg RMSE	Avg R^2	Avg max residual
Baseline	12.58	0.379	171
Linear regression	8.29	0.595	157.33
Ridge Regression (CV) (*α=0.0001*)	8.29	0.595	157.34
ARIMA	11.37	0.56	161.376
Random Forest (*max_depth=None, estimators=200, etc.*)	7.943	0.607	154.093
RNN	62.13	0.18	398.465

Average metrics are computed across year-folds. All models have optimized hyperparameters for each fold's predictions

Linear regression

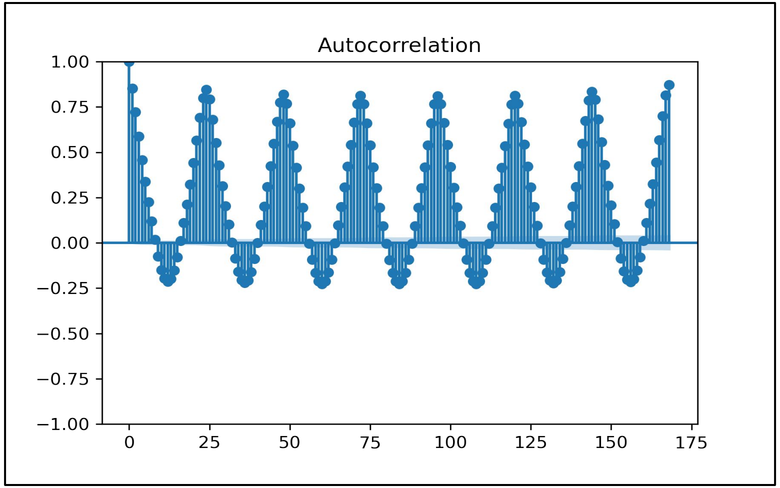
- Consistent underestimates in the test period – **heteroskedastic**
- Fairly accurate overall
- Perhaps lots of linear relationships in the data? Parametric model more suited to small (~8k) training data?

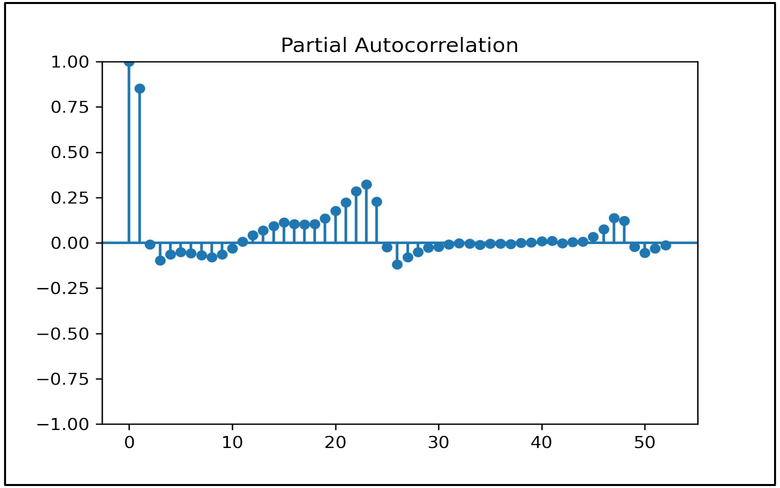


(80-20 train test split pictured for clarity)

ARIMA setup

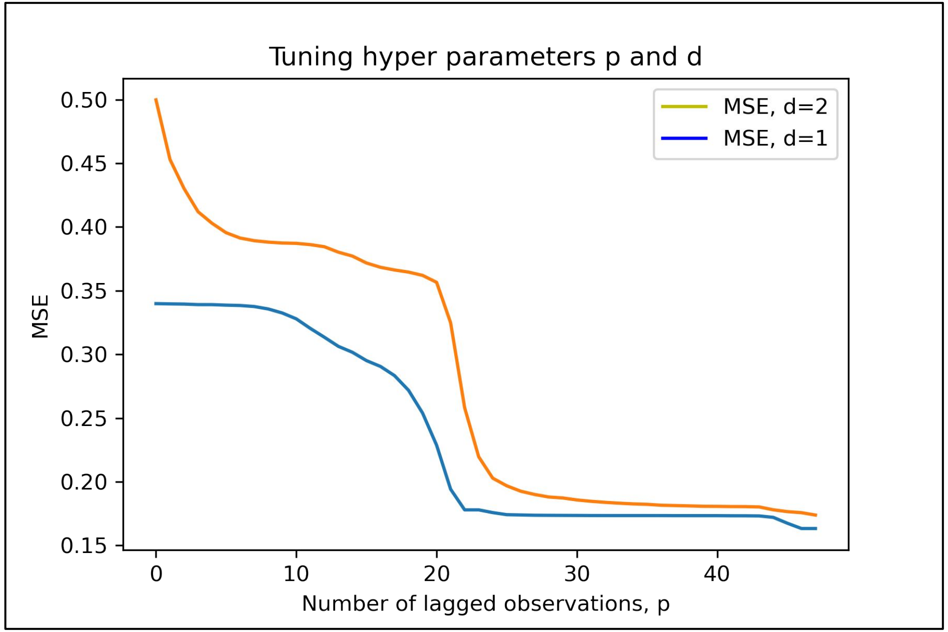
- Autocorrelation - significant hourly at the **week level** (and beyond)
- Partial autocorrelation - significant hourly up to about **2 days**
- Challenge: **intense computational requirements** as time lag increases
- Because of this, we **restricted our ARIMA model to 2 days**
- We **normalized the data** by each year and **clipped outliers** to be within 3 standard deviations





ARIMA hyperparameter tuning

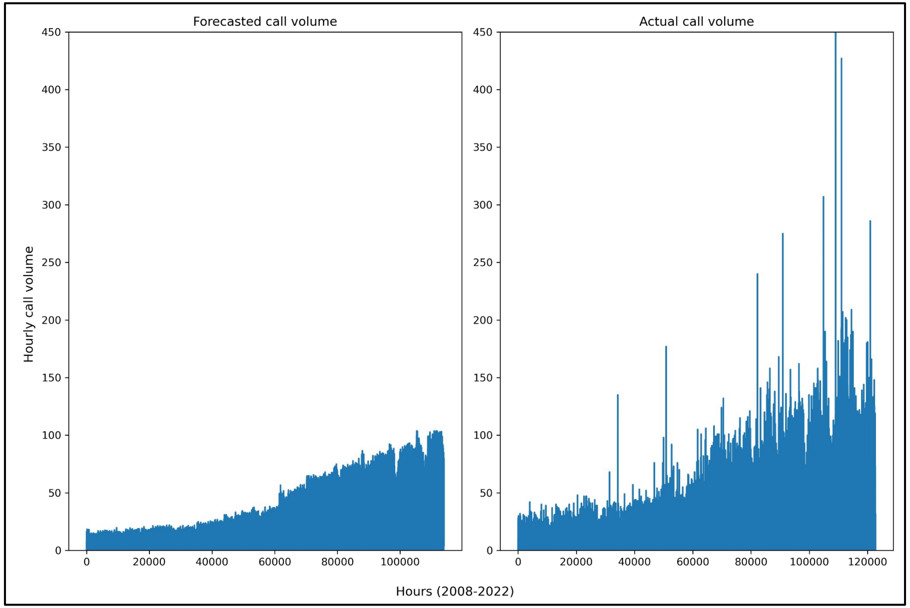
- Our model uses the **Auto Regressive (AR)** and **Integrative (I)** parts of the ARIMA model
- Because of this, we tuned the number of lagged observations (p) and the degree of differencing (d)
- We evaluated ARIMA models for **p=1** → **48** and **d=1** → **2**, evaluating RMSE and AIC

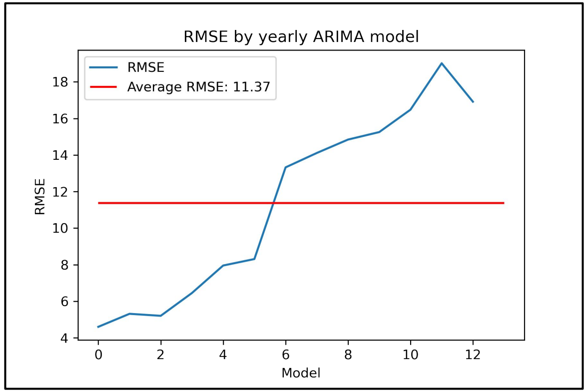


Note: RMSE graph reflects normalized data

ARIMA results

- Average RMSE across yearly models: **11.62**
- Positive trend in call volume data reflected in **heteroskedasticity of the error term**
- Normalization and outlier clipping improved RMSE but may exacerbate ARIMA issues with predicting volume spikes





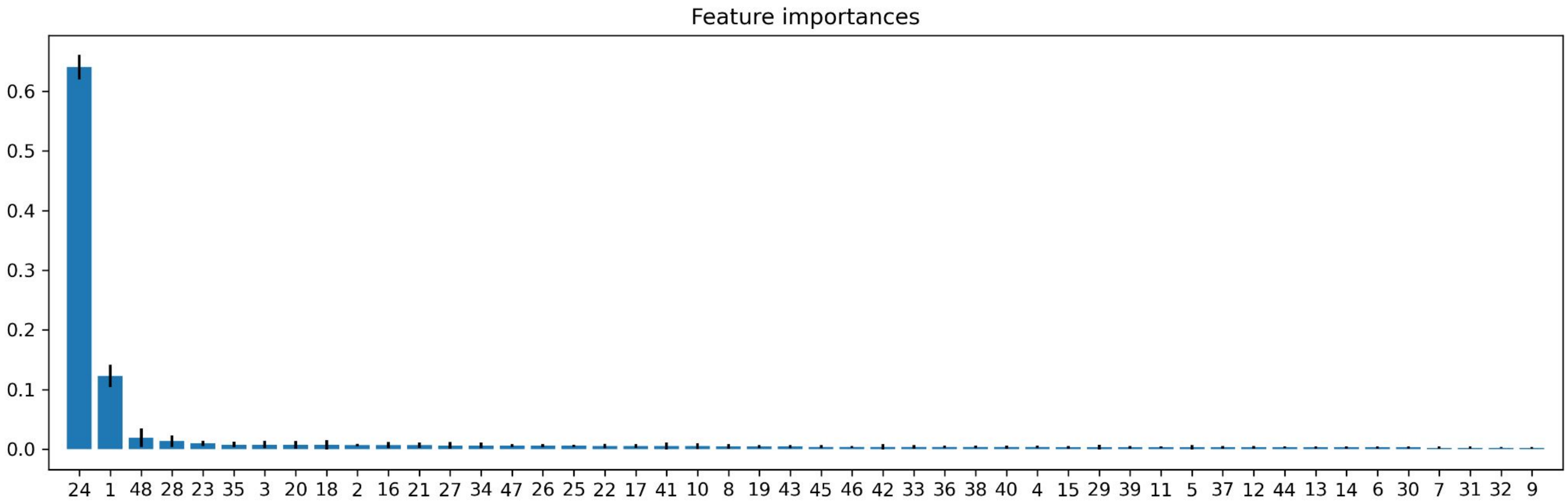
Recurrent Neural Network

- Tried LSTM, but hypothesized that data is too small, stuck to RNN
- Through hyperparameter exploration, we found that **deeper architectures were more effective**, but computationally costly – final RNN took 55 minutes to cross validate for **performance that was much weaker than simple linear regression**.

```
model = Sequential()
model.add(SimpleRNN(200,
activation='relu', return_sequences=True,
input_shape=(n_steps, 1)),)

model.add(Dropout(0.5))
model.add(SimpleRNN(units=150,
activation='relu', return_sequences=True))

model.add(Dropout(0.5))
model.add(SimpleRNN(units=100, activation='relu',
return_sequences=True))

model.add(Dropout(0.5))
model.add(SimpleRNN(units=50, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')
```

Random Forest Regression

- Our best model
- Easy to parallelize
- Focuses on similar features to the ARIMA



Next Steps

- Train optimal model at the census tract level (even smaller data) to let forecasting pick up geographic variation ⟹
- Further exploration of why RNN underperforms