Predicting Emergency Response Time in San Francisco

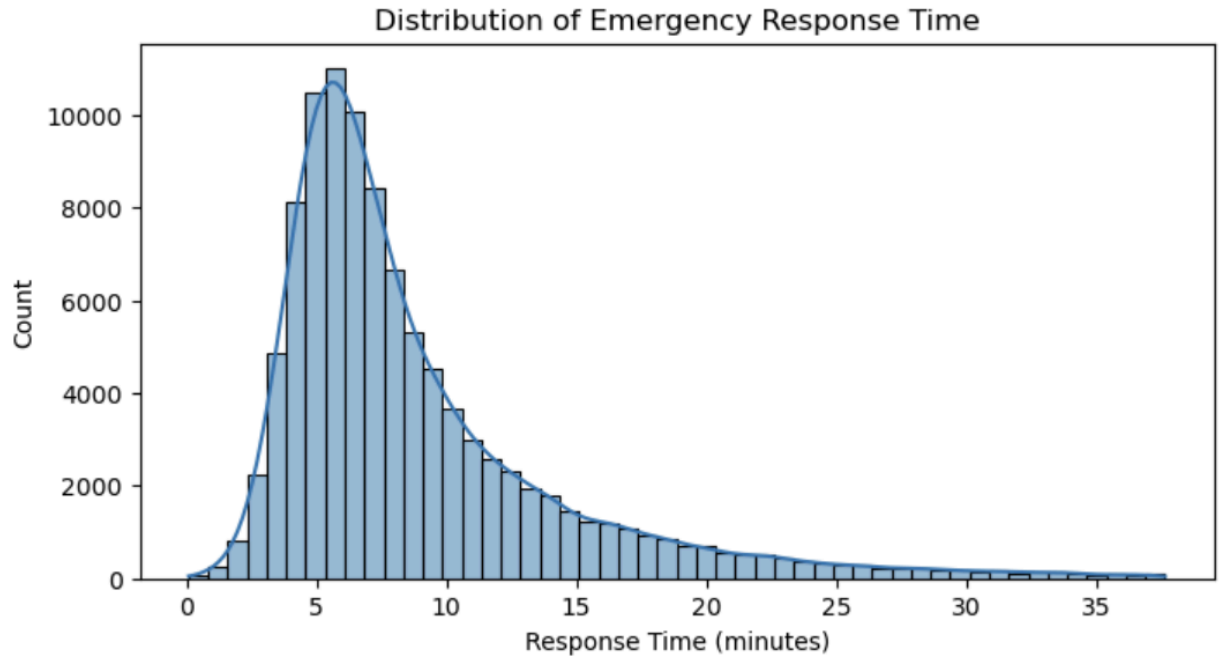
Harry Pachchala

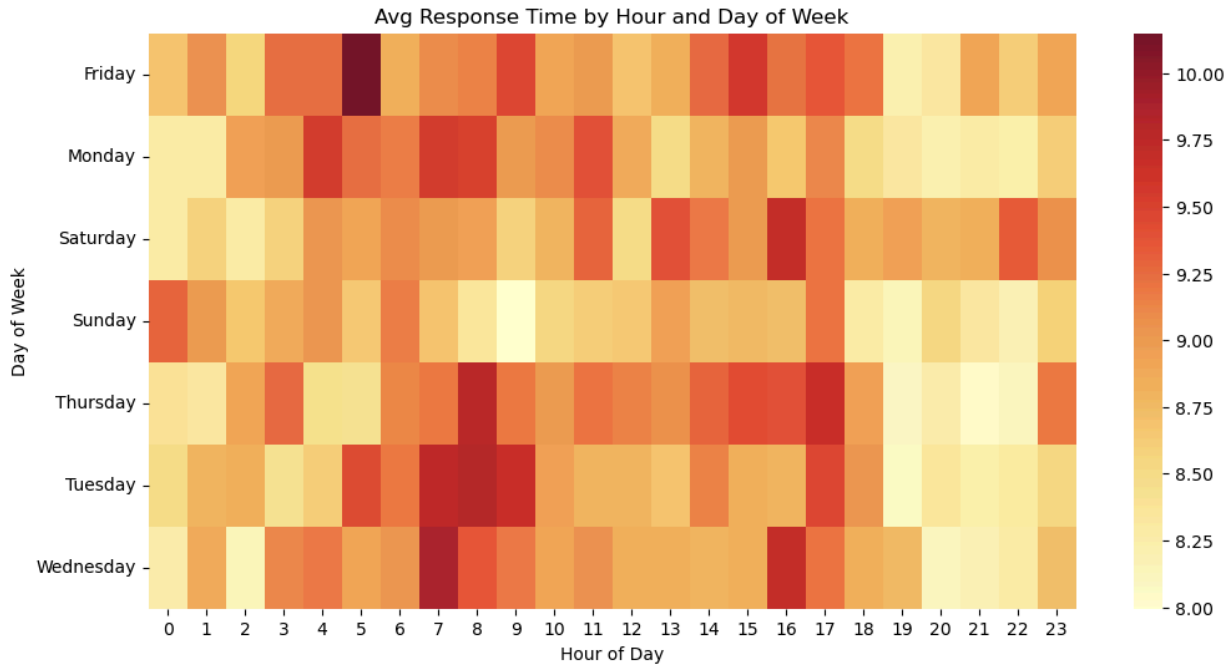
This project focuses on predicting emergency response time using public service call data from the San Francisco Fire Department. My goal was to model how long it takes a unit to arrive on scene after a 911 call is received, and to identify which factors such as call type, time of day, or unit dispatched most influence that duration. Emergency response time is a metric that cities and public agencies care about deeply, but it can be difficult to predict because it is influenced by a wide variety of contextual and operational variables. My approach was to use structured data that is already available to see how much we can uncover with the right features and models.

The dataset I used comes from the San Francisco Open Data Portal and includes over 130,000 rows of incident-level call logs. Each row represents a fire or medical unit dispatched to a specific event, along with timestamps for when the call was received, when the unit was dispatched, and when it arrived on scene. I used the difference between the received time and the on-scene time to calculate the target variable, which I called response_time. After filtering out nulls and removing outliers based on the 1st and 99th percentiles, I was left with a clean dataset of around 100,000 rows. I then extracted time-based features like hour, weekday, and month from the call timestamp. I also included categorical attributes like call type, unit type, station area, priority, and zip code.

I built a preprocessing pipeline using ColumnTransformer and Pipeline from scikit-learn. This allowed me to one-hot encode all categorical features, impute missing values, and keep my model inputs consistent across all experiments. I split the data into an 80/20 train and test set and built all models using this split. The majority of features in the dataset were categorical, but the model was still able to pick up meaningful patterns due to time-based variability and strong signals from features like call type and station area.

Before modeling, I performed exploratory data analysis to understand how response time varied by feature. The distribution of response time was right-skewed, with most responses falling between 5 and 10 minutes. Certain call types such as structure fires or elevator rescues had longer average response times, while medical calls were typically faster. I also found a clear time-of-day pattern, where late-night and early-morning calls had slightly higher response times, especially on weekends. Weekdays showed more stable averages. These insights supported my decision to include hour and day-of-week as predictors in the model. I included a heatmap showing how response time varied across different hours and weekdays to visually confirm some of these patterns

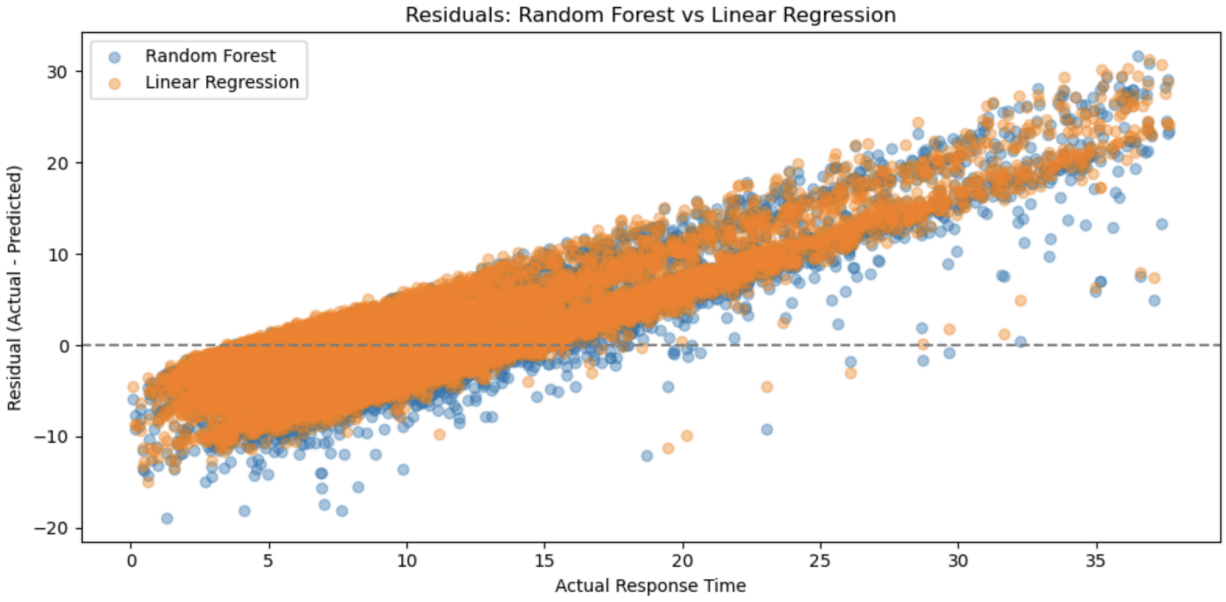




I treated this as a regression problem and tested a variety of models. I started with a dummy regressor to establish a baseline, followed by linear regression, ridge regression, random forest, gradient boosting, and a sampled k-nearest neighbors model. I evaluated each model using RMSE and R^2. The dummy regressor performed as expected, with an RMSE of about 5.69 minutes and an R^2 close to zero. Linear and ridge regression both achieved similar results, with RMSEs around 4.77 and R^2 values around 0.297. The random forest model improved slightly with an RMSE of 4.71 and R^2 of 0.315. Gradient boosting performed the best overall, achieving

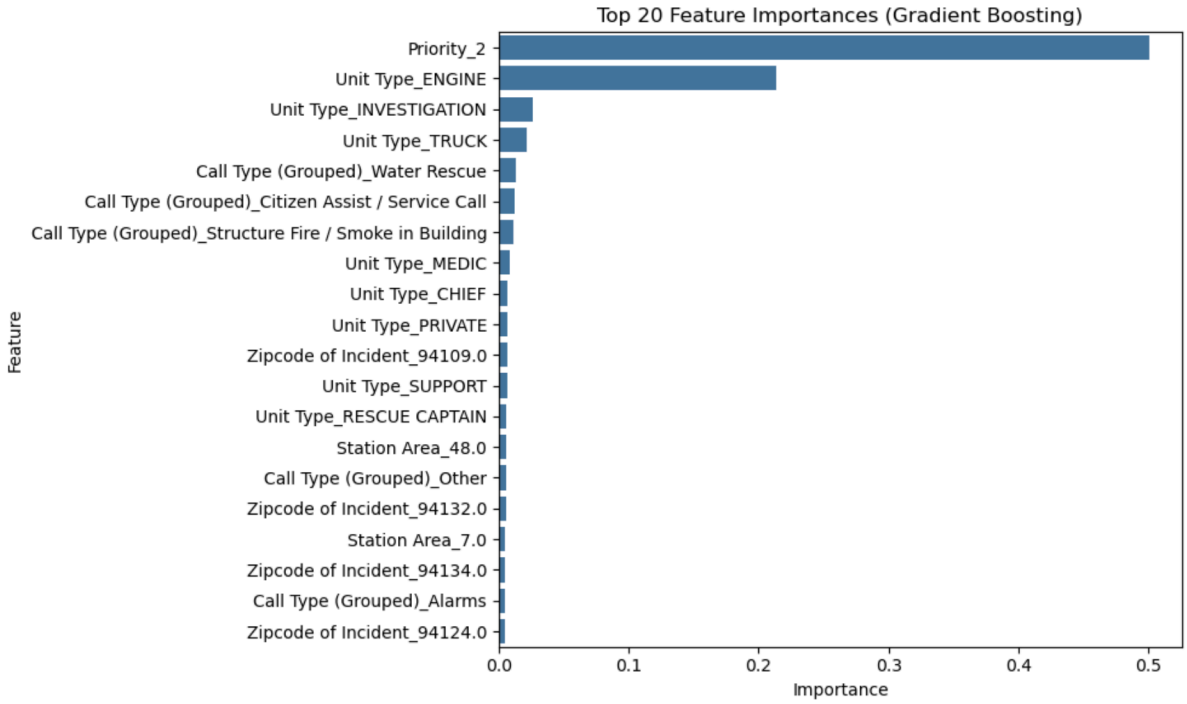
an RMSE of 4.68 and an R^2 of 0.324. While these are not high R^2 values, they clearly show that the models were picking up on meaningful structure in the data and outperforming the baseline. I included a summary table comparing the RMSE and R^2 values across all models. I also created a residual plot comparing Random Forest and Linear Regression predictions to analyze under and overestimation behavior.

	Model	RMSE	R2
0	Gradient Boosting	4.679381	0.324438
1	Random Forest	4.713074	0.314674
2	Ridge Regression	4.772730	0.297215
3	Linear Regression	4.773254	0.297061
4	KNN (sampled)	5.178146	0.172749
5	Dummy Regressor	5.693461	-0.000095



To help explain the model's predictions, I extracted feature importances from the gradient boosting model. The top predictors were dominated by priority and unit-related features rather than call type or time-based variables. The single most important feature was Priority_2, which significantly outweighed every other variable. This was followed by Unit Type_ENGINE, with all other features contributing only marginally to the model. Several unit types such as INVESTIGATION, TRUCK, and MEDIC appeared in the top 20, along with specific zip codes and station areas. Call type had very limited influence, with only a few categories like Water Rescue or Structure Fire showing up toward the bottom of the ranked list. This suggests that the

model relies more on operational attributes like priority and unit assignment than on call descriptions or timing. I included a bar plot visualizing the top 20 most important features based on the model's learned structure.



There are several limitations in this dataset and modeling approach. First, the dataset does not include contextual information like traffic, weather, or whether multiple calls were happening in the same area at the same time. These variables likely have a strong influence on response time but are not publicly available. Second, some of the features in the dataset, like priority or unit type, may not be used consistently across calls or units. There is also a limit to how much categorical features can explain something as variable as real-world dispatch behavior. Still, the model was able to extract useful patterns and reveal operational trends in how the city responds to emergencies.

While the overall R^2 of 0.324 is modest, it is meaningful in the context of real-world public operations data. This project shows that structured call data can be used to detect consistent patterns in response time, even without access to more complex variables like location tracking or dispatch queue data. In the future, this model could be used to flag unusually long response times, predict delays based on known call types or locations, or help departments allocate resources more efficiently during high-risk time windows. If I had more time, I would explore classification framing such as predicting whether a response will exceed 10 minutes, and I would integrate external data sources like real-time traffic. I would also consider exploring

interaction effects between features such as call type and time of day to see if certain combinations are especially prone to delays.