**An analysis of factors influencing Get It Done San Diego request resolution time**

Marissa Westerfield

In 2015, the City of San Diego conducted a city-wide survey to determine what services residents felt were most important for the city to provide, and how satisfied residents currently were with those services[[1]](#footnote-1). Survey results were that infrastructure maintenance was one of the most important services to residents, second only to police services. When combined with satisfaction ratings, the city identified infrastructure maintenance as a target for increased emphasis because while that service was of high importance to residents, it was underperforming customer expectations.

In 2016, as part of an ongoing effort to improve customer satisfaction, the City of San Diego launched the Get It Done mobile app (and related page on the city government website). Residents could use this app to report issues such as broken streetlights, needed sidewalk repairs, and storm drains, as well as nuisances such as parking violations and trash cans left on the street. Since its launch, over 300,000 requests for service have been logged; all requests are publically available on the City of San Diego’s Open Data Portal[[2]](#footnote-2).

The City of San Diego is divided into nine districts (see Fig. 1). District 8 is notable for being divided by the City of Chula Vista (a portion of it is contiguous with Districts 3, 9, and 4). The denser, urban areas are in Districts 3, 8, 9; the other districts are suburban with varying degrees of density. In order to monitor the City’s performance, it is important to know not only how successful overall the City has been at resolving requests made by its residents, but also if the City has allocated its resources equitably across all of its regions. The answer to this question may aid the City in decisions to reallocate resources, if needed.

This report provides a summary of the entire project. It describes the data wrangling procedure, some initial findings from the data, some exploratory data analysis results, and the results of a machine learning model to predict the amount of time it takes to close requests. All code related to this project is available at: https://github.com/mwesterfield/Portfolio/get-it-done-sd-analysis

***Fig. 1***. *District map for the City of San Diego[[3]](#footnote-3)*

**Data Wrangling**

The Get It Done (GID) data is downloadable as a CSV file. It was imported into a pandas dataframe, with the following fields used for further preprocessing and analysis (see Table 1).

**Table 1**. Original GID fields retained for preprocessing and analysis

|  |  |
| --- | --- |
| **FIELD** | **DESCRIPTION** |
| service\_request\_id | unique identifier for each request |
| requested\_datetime | date and time the service request was made |
| service\_name | name of service request type |
| case\_record\_type | indicates City staff group responsible for responding to the request |
| updated\_datetime | date and time the request was last modified or closed/referred |
| status | current state of service request |
| lat | latitude for request location |
| long | longitude for request location |
| district | Council district of request location |
| case\_origin | how the request was submitted |

Several additional variables were created from requested\_datetime and updated\_datetime values (all new fields shown in Table 2, next page). For requests with status set to ‘closed,’ the difference between the updated\_datetime and requested\_datetime was calculated, then converted to a float representing the number of days that passed until the request was closed (‘days\_till\_closed’). Also, binary variables were created for a categorical representation of ‘days\_till\_closed’. A request could be ‘closed\_in\_14days’ (within the median amount of time a request is open), ‘closed\_in\_45days’ (within the first three quartiles of time to close), or ‘closed\_after\_60days’ (the upper fence of time to close).

Two more variables were created to track the number of unresolved requests in each service\_name category at a specific point in time. For an individual request, the total number of unresolved requests (at that datetime) in that same service\_name category was logged. The total number of unresolved requests in the same service\_name category in a single zipcode was also logged.

Demographic information was added to the dataframe in a two-step procedure. First, the zipcode corresponding to the location of each request was estimated using the USZIPCODE API[[4]](#footnote-4) and added to the dataframe (for every individual location, the zipcode search started within a 1-mile radius; if unsuccessful, was increased by 1-mile steps to a maximum of three miles). Next, zipcode-specific demographic information—median housing value, median household income, population density, and median age—was added to the dataframe. 2010 Census data with housing value, household income and median age separated by zipcode is available from the SANDAG Data Surfer site[[5]](#footnote-5). Population density within each zipcode (also

**Table 2**. New fields added to the GID dataframe

|  |  |
| --- | --- |
| **FIELD** | **DESCRIPTION** |
| days\_till\_closed | amount of time, in days, from request submission to closure |
| closed\_in\_14days | 1 if days\_till\_ closed < 15; 0 otherwise |
| closed\_in\_45days | 1 if days\_till\_closed < 46; 0 otherwise |
| closed\_after\_60days | 1 if days\_till\_closed > 59; 0 otherwise |
| resolved | 1 is status is (‘Closed’ or ‘Closed – Referred’); 0 otherwise |
| zipcode | zipcode of request submission, based on lat/long information |
| household\_income | median household income in request’s zipcode |
| housing\_value | median housing value in request’s zipcode |
| median\_age | median age of population in request’s zipcode |
| pop\_density | population density (per XX) in request’s zipcode |
| load\_by\_service | total number of unresolved requests in the same service\_name category at the time the request is submitted |
| load\_by\_service\_zip | total number of unresolved requests in the same service\_name category and zipcode at the time the request is submitted |

based on the 2010 Census) was downloaded from the The Splitwise Blog[[6]](#footnote-6).

Over the past several months, the City of San Diego has refined the structure of the Get It Done datasets, probably to stay current with Get It Done app updates, and possibly as City analysts determine which information in the database is truly useful. For example, it appears that many of the ‘service\_name’ categories have been dropped after a major update to the mobile app in July 2018. Also, the total number of fields provided in the downloadable spreadsheet has been reduced from 29 fields available in July 2018 to 12 fields as of January 2019.

For the most part, this project has been unaffected by these changes—the fields used for these analyses are still provided. One area where this change may have a real--and potentially negative—impact is that the database no longer appears to track duplicate requests. The spreadsheet available in July 2018 had three separate fields dedicated to tracking duplicate requests (‘parent\_case’, ‘duplicate\_verified’, ‘override\_duplicate’). There was also a ‘Duplicate’ category in the ‘status\_description’ field. The spreadsheet available in Nov. 2018 no longer contained the duplicate-specific fields, although ‘Duplicate’ was still a category in ‘status\_description’. The most recent spreadsheet (available Jan, 2019) has no information whatsoever about duplicate submissions, although the requests that were previously flagged as duplicates are still included in the spreadsheet. The number of duplicate responses in the database is large enough, if erroneous, to affect analyses. Please see Appendix A (“Analysis of Duplicate Requests”) for a more detailed examination of duplicate requests. To the extent possible, duplicate requests were removed from the dataset during preprocessing, and were not included in the analyses presented in this project.

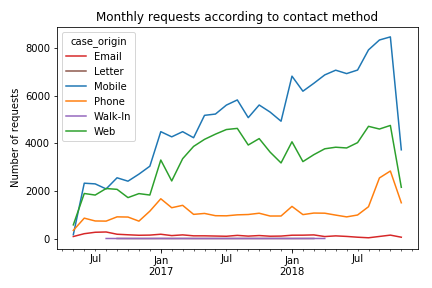
**Exploratory Data Analysis**

Exploratory data analyses revealed several interesting features that will be discussed in separate sections below. The first few sections focus on requests made by San Diego residents only; requests are made by City employees themselves (as the result an internal report or referral or when out on a maintenance run) have been discarded from these analyses. Later sections include both resident-submitted and employee-submitted requests.

How do San Diegans communicate with their City?

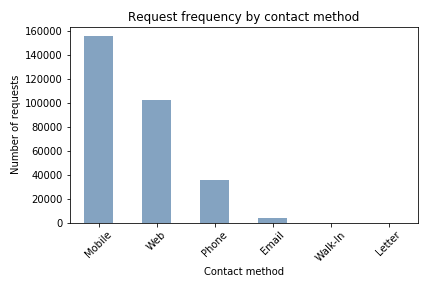
*Have San Diego residents adopted the Get It Done mobile app?*

By tracking the number of requests made per month since the app’s introduction in May 2016, we can see that mobile app usage has increased steadily (see Figure 2). Requests made via the City website also increased initially, but appear to have plateaued over the past year and a half. The number of requests made by phone has been largely stable; email requests, in contrast, have declined over time. Overall, San Diegans appear to have adopted the Get It Done mobile app as a useful means of communication with the City, although web and phone access remain important lines of communication.

***Fig. 2*** *Mobile app usage has increased steadily over time. Website usage has increased more slowly, but other means of contact have not shown those gains. The drop at the end of the graph is due to having only partial data for Nov. 2018*

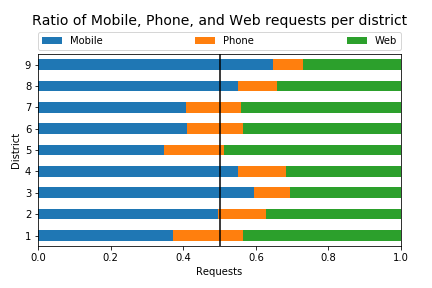
*Communication preferences over San Diego as a whole*

Overall, most requests are submitted via the mobile phone app (153,936 or 52.1%) or online through the City website (102,109 or 34.6%). A sizeable number of requests are made over the phone (35,282 or 11.9%), and a much smaller number are made by email (3,967 or 1.3%). Fewer than 20 requests were made by letter or by walk-in combined (< 1%), see Figure 3.



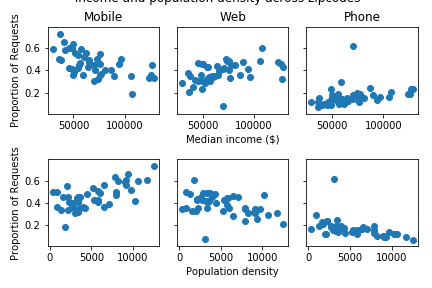
***Fig. 3*** *The majority of service requests made by San Diego residents are submitted via the mobile app. Web submissions, phone calls, and to a smaller extent email, are also popular ways to communicate with the City*

*Communication preferences broken down by City District and zipcode*

****When service requests are further categorized by district, it is evident that the mobile app is the preferred communication method for half of San Diego’s districts (e.g., Districts 2, 3, 4, 8, and 9), web access is preferred in Districts 1 and 5, and Districts 6 and 7 are more balanced between mobile and web access (see Figure 4).

***Fig. 4*** *Each bar shows the relative frequency of requests made either through the mobile app, by phone call, or through the City website in each individual district. The midpoint is marked by the black vertical line.*

By combining demographic information (obtained from the 2010 Census) per district with mobile app, website, or phone usage rate, it appears that both household income and population density predicts which communication method will be more popular within a zipcode. More dense, lower income neighborhoods are more likely to use the mobile app, whereas less dense, more wealthy neighborhoods are more likely to use the City website or to call the City directly.



***Fig. 5 (top row)*** *Median household income per zipcode is plotted against rate of use of the three most popular communication methods.* ***(bottom row)*** *Population density in each zipcode is plotted against rate of use of communication method.*

Followup correlation testing and linear regression models were also tested in order to identify the most important demographic predictors for each communication type.

*Mobile app*: The percentage of requests submitted by mobile app was significantly correlated with population density (r = .72, p < 0.01), median income (r = -.65, p < 0.01), median age (r = -.45, p < 0.01), and housing value (r = -.34, p = 0.02). When fitting those variables to a multiple regression model, the predictors explained 57.4% of the variance (R2 = .57, F(4,39) = 15.48, p < 0.01) Household income (β = -1.75x10-6, p = 0.03) and population density (β = 1.63x10-5, p < 0.01) significantly predicted mobile app use.

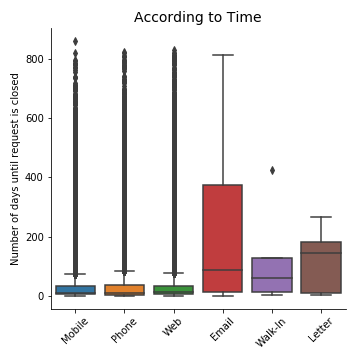
*Phone*: The percentage of requests submitted over the phone was significantly correlated with population density (r = -.47, p < 0.01). When fitting all variables to a multiple regression model, the predictors explained 18.9% of the variance (R2 = .18, F(4,39) = 3.49, p = 0.02) Only population density (β = -1.58x10-5, p = 0.01) significantly predicted phone use.

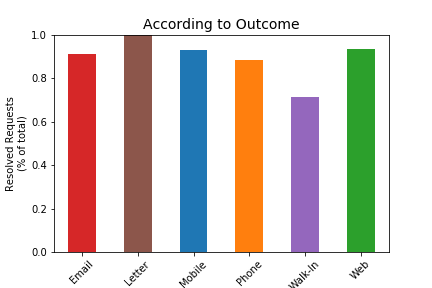
*Website*: The percentage of requests submitted via website was significantly correlated with household income (0.48, p < 0.01) and population density (r = -.36, p = 0.02). When fitting all variables to a multiple regression model, the predictors explained 17.7% of the variance (R2 = .17, F(4,39) = 3.31, p = 0.02) Only household income (β = 2.14x10-6, p = 0.02) significantly predicted website use.

Effects of communication method on the City response

*What is the most effective way for a San Diegan to communicate with the City?*

The percentage of service requests that have been resolved (marked either ‘Closed’ or ‘Closed – Referred’) and the amount of time it takes to close a service request are shown in Figure 6. If successful resolution of a request is the primary measure of effectiveness, it appears that residents are most successful when they submit requests over the mobile app (92.8%) or or City website (93.4%). Requests submitted via email (90.1%) and over the phone (88.2%) were less effective. The difference in resolution rate was significant across email, mobile, phone, and

 Communication Effectiveness



0.91

0.93

0.88

0.93

0.71

***Fig. 6*** ***(left)*** *Percentage of total requests that are successfully resolved as a function of communication method;* ***(right)*** *length of time (in days) it takes to close a request as a function of communication method*

web (χ2(1, N=292,743) = 1072.9, p < 0.001). Requests made by letter or walk-in were not included in this analysis since there were fewer than 20 of these. Of the requests that are successfully closed, communication via mobile app, phone, and website have the fastest closure time, with a median closure time of approximately 14 days.

The distribution of time-to-close for all closed requests is massively skewed right, which may obscure patterns in the data, therefore I take two approaches to examining the City response to requests: 1) by calculating the “timely-closure rate,” which is the percentage of requests that are closed within 60 days (60 days is the upper fence of the “days\_till\_closed” distribution); 2) by calculating the mean “days\_till\_closed” of requests that are closed within 60 days.

*Is timely-closure rate different according to request method?*

The test of homogeneity of proportions was used to determine whether timely-closure rate differed according to request method. The first comparison was between all requests submitted by City employees and all requests submitted by San Diego residents. Timely closure rates were significantly different (χ2(1, N=179,671) = 295.44, p < 0.001), with a lower closure rate for requests submitted by San Diego residents (83.1%) than by employees (89.0%).

A second comparison was between only requests from the public submitted with the mobile app, over the phone, and through the City website. While there was a statistically significant difference in timely-closure rate (χ2(2, N=163,628) = 202.8, p < 0.001, there was not much practical difference between mobile app (84.5%), phone (81.4%), and website (83.9%).

*Is the amount of time to close a request different according to request method?*

This analysis looked at only requests from the public submitted via mobile app, phone, or website. A one-way ANOVA showed a significant difference in the time needed to close a request between mobile app (mean = 13.5 days), phone (mean = 11.6 days), and web requests (mean = 14.5 days), F(2, 132775) = 358.2, p < 0.001.

Other factors that may impact the City response

*Is timely-closure rate related to demographics?*

The overall percentage of requests closed within 60 days was correlated with household income (r = -.66, p < 0.001), population density (r = .61, p < 0.001), housing value (r = -.50, p < 0.001), and median age (r = -.42, p < 0.01). When fitting those variables to a multiple regression model, the predictors explained 43.3% of the variance (R2 = .43, F(4,39) = 9.21, p < 0.001) However none of the predictors were significantly predictors of timely-closure rate. Some of these demographic measures are highly collinear however; a multiple regression model using only household income and population density explained approximately the same amount of variance (R2 = .46, F(4,39) = 19.15, p < 0.001), and in this case household income (β = -8.25x10-7, p < 0.01) significantly predicted timely-closure rate.

**Machine Learning – How long will it take to close a request?**

The primary goal of the machine learning approach was to create a model that can predict how long it will take to close any request submitted to the City. A secondary goal was to use the resultant model to derive additional insights about how the City resolves requests.

Out-of-the-box versions of multiple models were tested to get a feel for the type of model that would perform the best with the Get it Done data (code can be found in Notebook4a-Machine\_Learning\_Part1.ipynb). Because the goal was to predict a numeric value (number of days it takes to close a request), I initially tried regression models (linear regression and random forest regressor), and then moved to classification models because performance was not acceptable. The regression models attempted to predict how many days it would take to close an individual request; the classification models attempted to predict which category of closure time a request would fall into (categories were: ‘closed in 5 days or less’, ‘closed between 6 days-1 month’, ‘closed between 1-2 months’, and ‘closed after 2 months’).

*Regression models*

R2 was used as the performance metric for both the linear regression and random forest regressor models. As shown in Table 1, test set performance for both models was abysmal, and there is evidence that the random forest regressor overfit the data. The extremely poor performance of the linear regression model is an indication that the data failed to meet data distribution assumptions—highly likely in this case because the distribution of days\_till\_closed (the target variable) is massively skewed to the right. Because of this, multilabel classification (tree) models would be a more appropriate choice.

*Multilabel classification models*

For these models, accuracy (percentage of correctly predicted requests) and (unweighted) f1 score were both used to rate performance. The f1 score was included because it provides additional information about miscategorization (it includes false negatives and false positives in the score along with true positives). In this case, the average=’macro’ option was used, which affords more weight to the lower-represented classes, so the f1 score is more negatively impacted by misclassification in under-represented classes, whereas the accuracy score is more

**Table 3: OOTB Performance of all models tested**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Regression models | | | | |
|  | Training set R2 | | Test set R2 | |
| Linear regression | 0.417 | | -4.357 x 1017 | |
| Random Forest Regressor | 0.972 | | -0.020 | |
| Multilabel classification models | | | | |
|  | Training set accuracy | Test set accuracy | | Test set f1 score |
| Random Forest | 0.987 | 0.547 | | 0.492 |
| Naïve Bayes | 0.457 | 0.448 | | 0.417 |
| Gradient Boost | 0.516 | 0.493 | | 0.470 |
| ADA-Boost | 0.469 | 0.446 | | 0.434 |

negatively impacted by misclassification in over-represented classes. Looking at both scores thus provides a more complete picture of performance than looking at only one measure.

The highest scores are highlighted in Table 3 (above), and indicate that overall, the random forest classifier performs the best with the test data. This one model is still prone to overfitting (as was the random forest regressor), but test set performance is good enough to indicate that it should be the model chosen.

*Hyperparameter tuning*

After selecting the random forest multilabel classifier, a series of parameter tunings was performed to find the best hyperparameters for the model. Table 4 (below) shows the hyperparameters after each stage of tuning.

**Table 4: Hyperparameters**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Out of the box | Coarse random search | Fine grid search |
| n\_estimators | 10 | 1600 | 1600 |
| min\_samples\_split | 2 | 2 | 2 |
| min\_samples\_leaf | 1 | 1 | 1 |
| max\_features | ‘auto’ | ‘sqrt’ | 15 |
| max\_depth | None | 40 | 45 |
| Bootstrap | True | True | True |

*Final model fitting and performance*

Fitting the model to the fully upsampled dataset using the hyperparameters found in Part 2 results in an overall test set accuracy of 0.582 and f1 score of 0.525. Random forest models are not as simple to interpret as regression models, but using both accuracy scores and the confusion matrix to visualize performance can provide a bit more information about what in the dataset is actually predictable versus what is not. Table 5 (below) shows model performance for a subset of those service categories—the ten most frequently requested services by San Diegans. We can see from Table 5 that the model performs inconsistently across these service categories. The low f1 scores imply that for service categories where the model performs with high accuracy, this accuracy is driven by correct predictions for the majority class. So that the very least, the model has learned general time-to-close characteristics of the various services offered by the City.

As an example, see the confusion matrix plots for three frequently requested services: storm water code enforcement, 72 hour violation, and traffic sign maintenance (Figure 6, below). The plots for storm water code enforcement (Figure 6, top row) show that when an overwhelming number of requests fall into the same time-to-close bin (longer than 2 months, in this case), the model will tend to make that same prediction for all requests in that service category. For 72 hour violation requests (Figure 6, middle row), the overwhelming number of requests fall into the first two bins (within 5 days and between 5 days to 1 month), but the

**Table 5**: Model performance on services most-frequently requested by San Diego residents

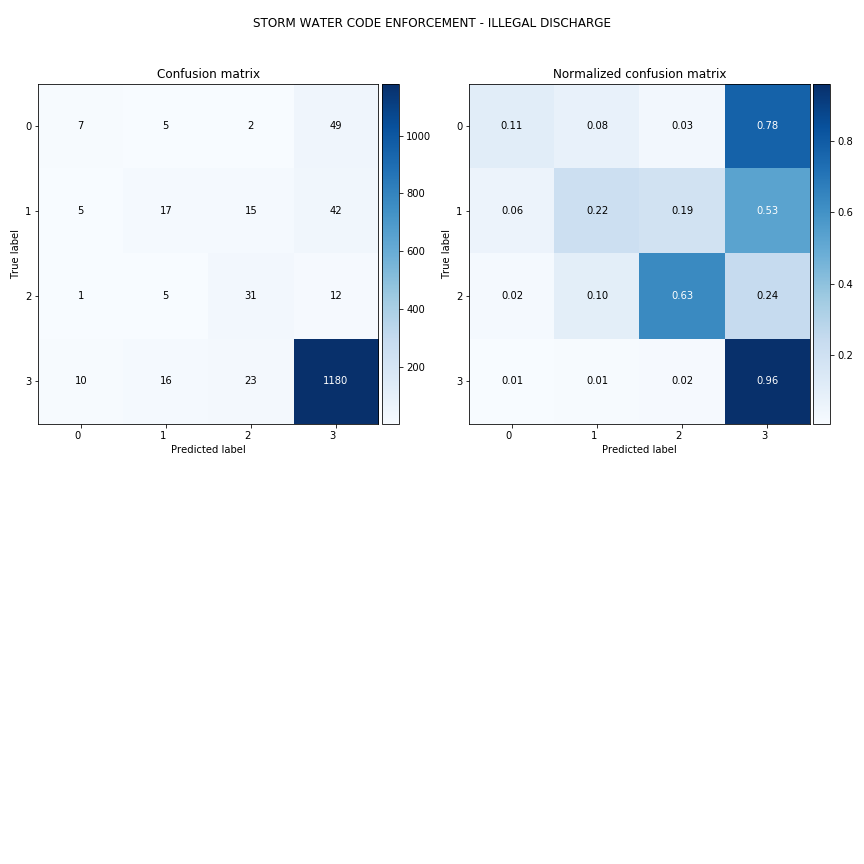
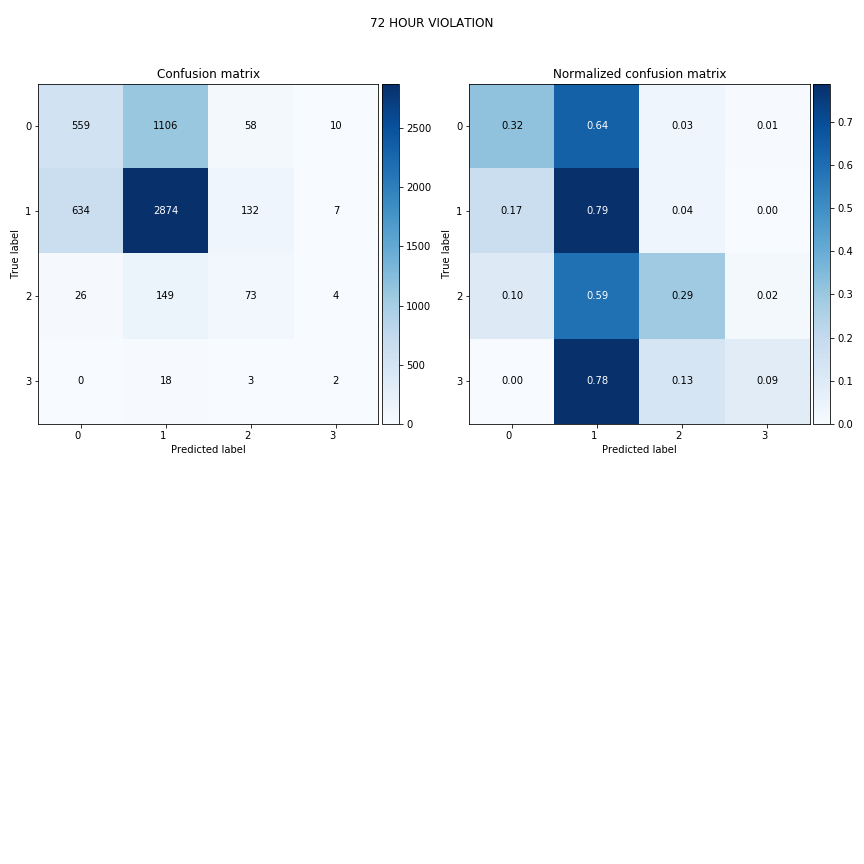
|  |  |  |
| --- | --- | --- |
| **service\_name** | **accuracy** | **f1 score** |
| STORM WATER CODE ENFORCEMENT - ILLEGAL DISCHARGE | 0.871 | 0.476 |
| 72 HOUR VIOLATION | 0.620 | 0.371 |
| GRAFFITI REMOVAL | 0.611 | 0.473 |
| ILLEGAL DUMPING | 0.590 | 0.464 |
| POTHOLE | 0.555 | 0.478 |
| TRAFFIC SIGNAL LIGHT OUT | 0.545 | 0.416 |
| SIDEWALK REPAIR ISSUE | 0.522 | 0.460 |
| STREET LIGHT OUT | 0.495 | 0.409 |
| PAINT CURB - MAINTAIN | 0.493 | 0.322 |
| TRAFFIC SIGN - MAINTAIN | 0.415 | 0.417 |

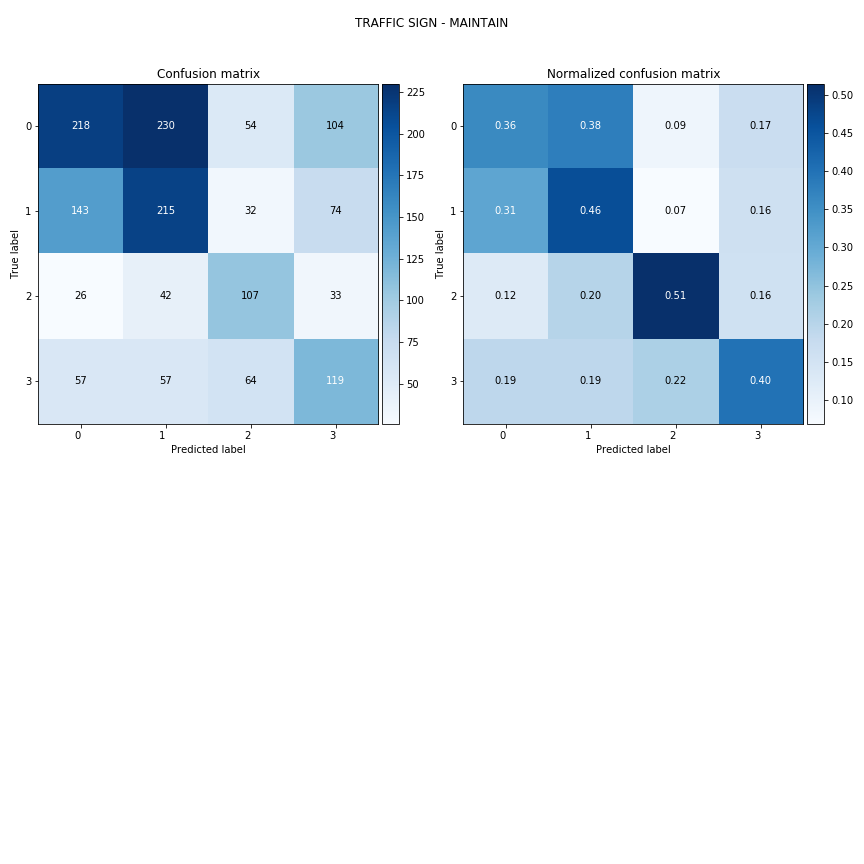
model still tends to predict a single bin (5 days to 1 month) for most of the requests. For traffic sign maintenance (Figure 6, bottom row), where closure times are less tightly clustered, the model predicts that dispersion with modest success indicated by the darker blue colors along the diagonal in the normalized confusion matrix plot.

We can use confusion matrix plots to get more information about other predictors as well. Figure 7 (below) shows confusion matrices for requests categorized according to method used to submit the request to the City (e.g. email, mobile app, phone, or website). These plots show that the model performs best at predicting time-to-close for requests submitted over the mobile app or by phone, and has more trouble with predicting time-to-close for requests submitted through the City website or email.

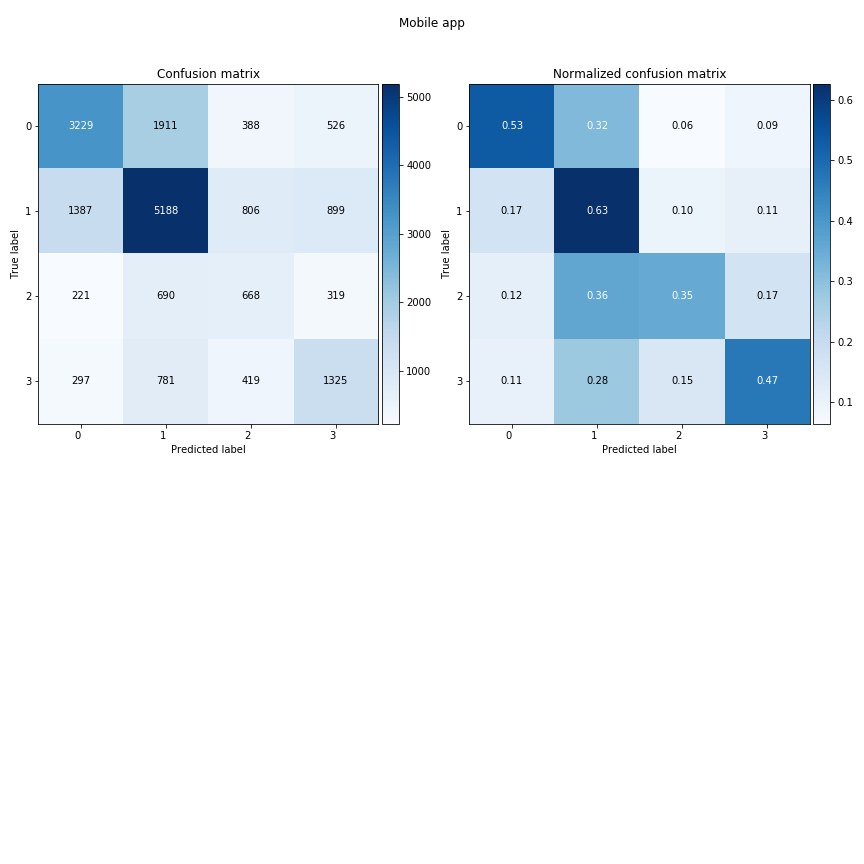
Finally, Figures 8 (standard confusion matrices) and 9 (normalized confusion matrices) show that there are not major differences in model performance according to City District. This is an encouraging result because it ties back to one of the original questions of this project, which is whether resources are equitably distributed across City regions. That the model does not show performance differences based on district is an indication that services are provided in as equitable manner as possible.

It is otherwise difficult to get an intuitive feel for how this model performs. Load\_by\_service (the number of open requests across the City in that same service category at the time the request is submitted) and Load\_by\_service\_zip (open requests in the same service category and in the same zipcode) are the strongest features in the model. Household income and population density are also among the top predictive features, but the fact that the model has a max\_depth parameter of 45 and n\_estimators parameter of 1600 indicates that this is a highly complex model that resists easy interpretation.

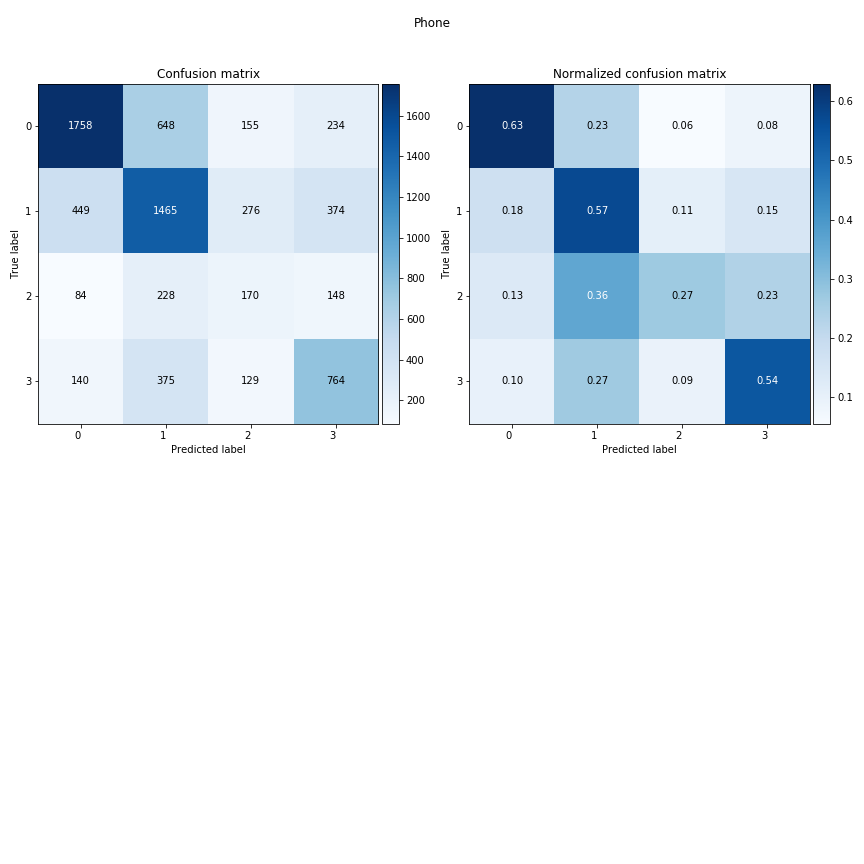




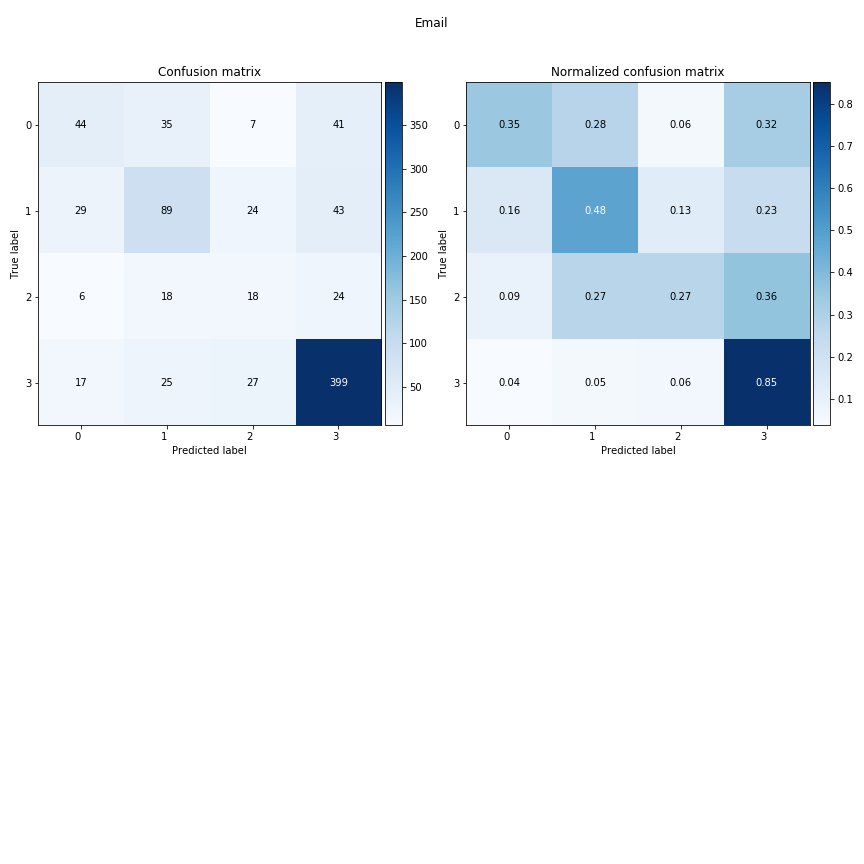
**Figure 6**: Confusion matrices for three frequently requested services, storm water code enforcement (top), 72 hour violation (middle), and traffic sign maintenance (bottom). Raw service request numbers are displayed in the left column of plots (true label vs. predicted label); values normalized across true labels are displayed in the right column. Darker blue corresponds to better predictions. Bin labels are: 0 = request is closed within 5 days; 1 = closed after 5 days but before 1 month; 2 = closed between 1-2 months; 3 = closed after 2 months

Mobile App

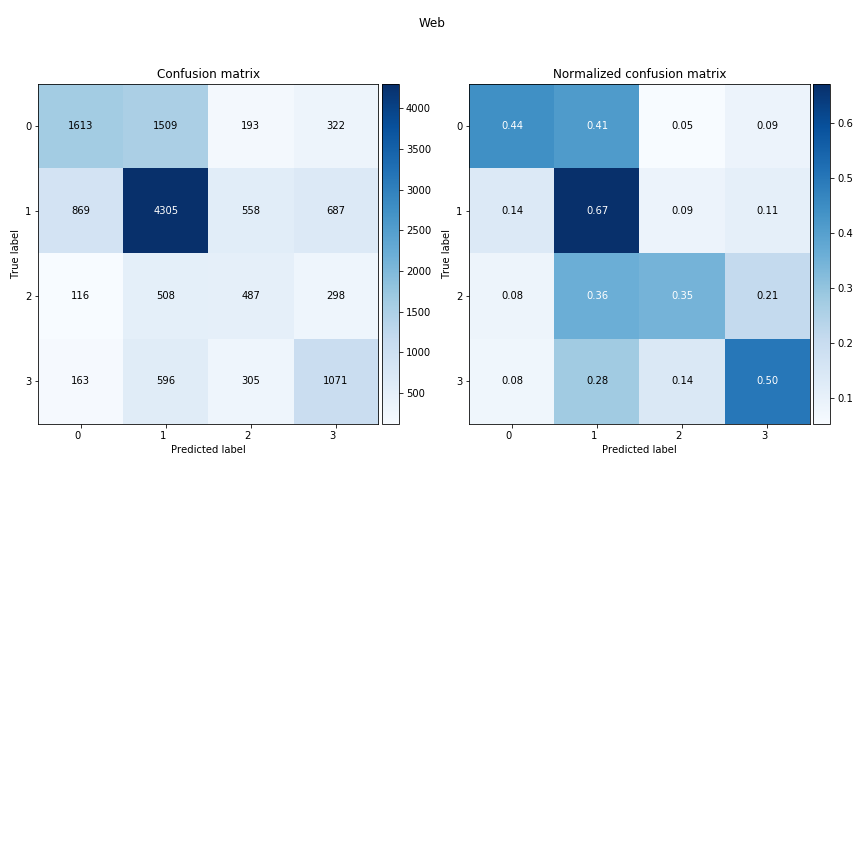
Phone



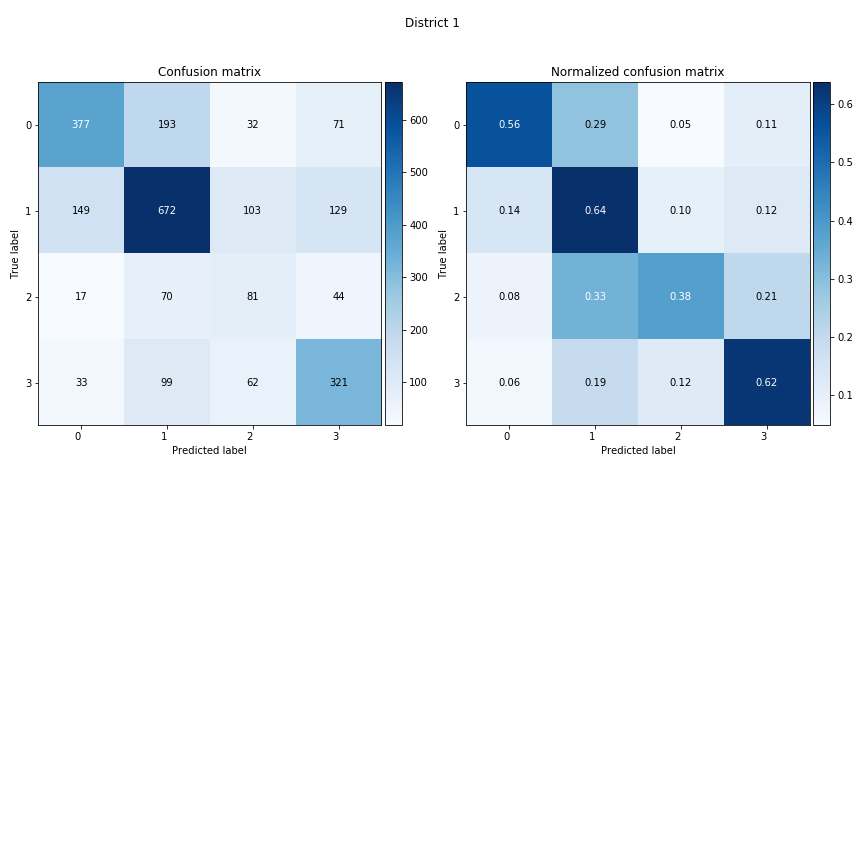
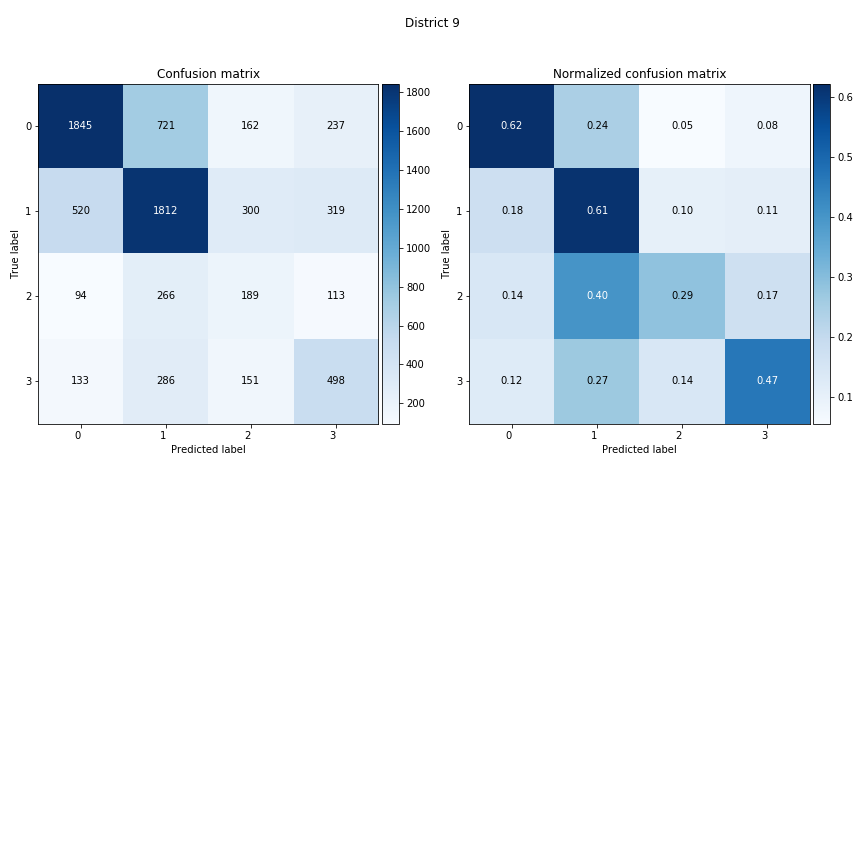
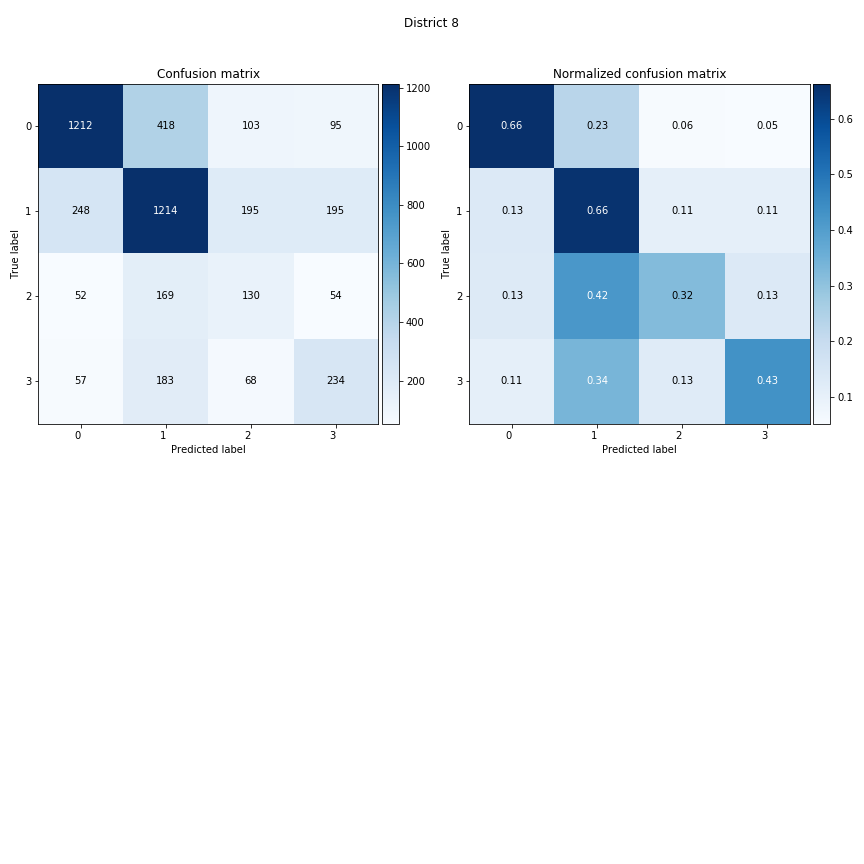
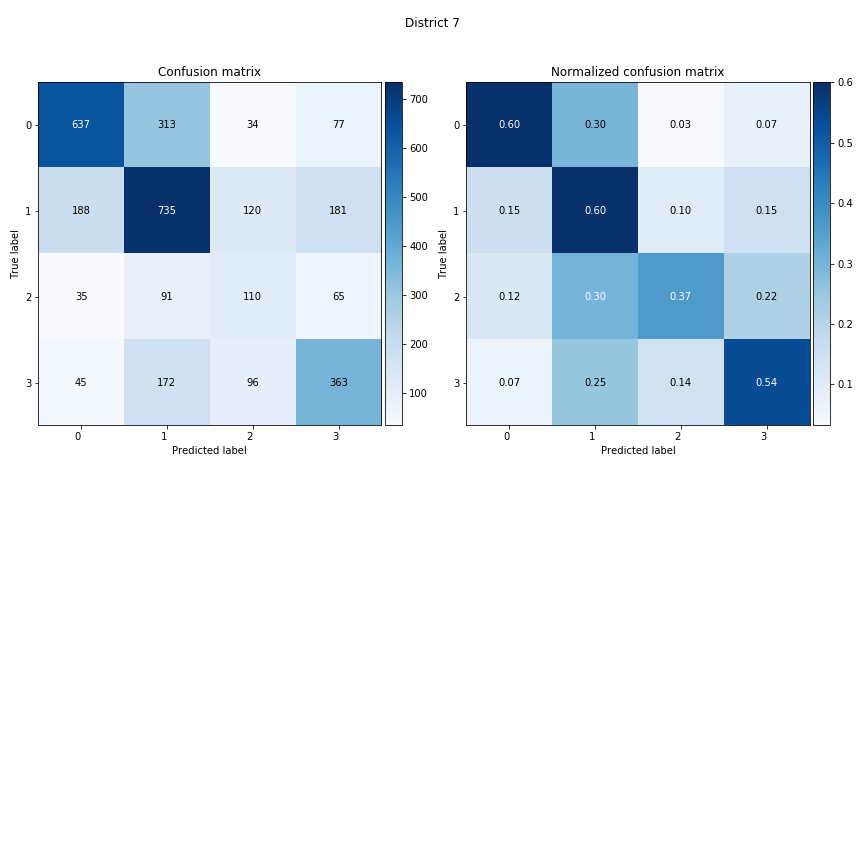
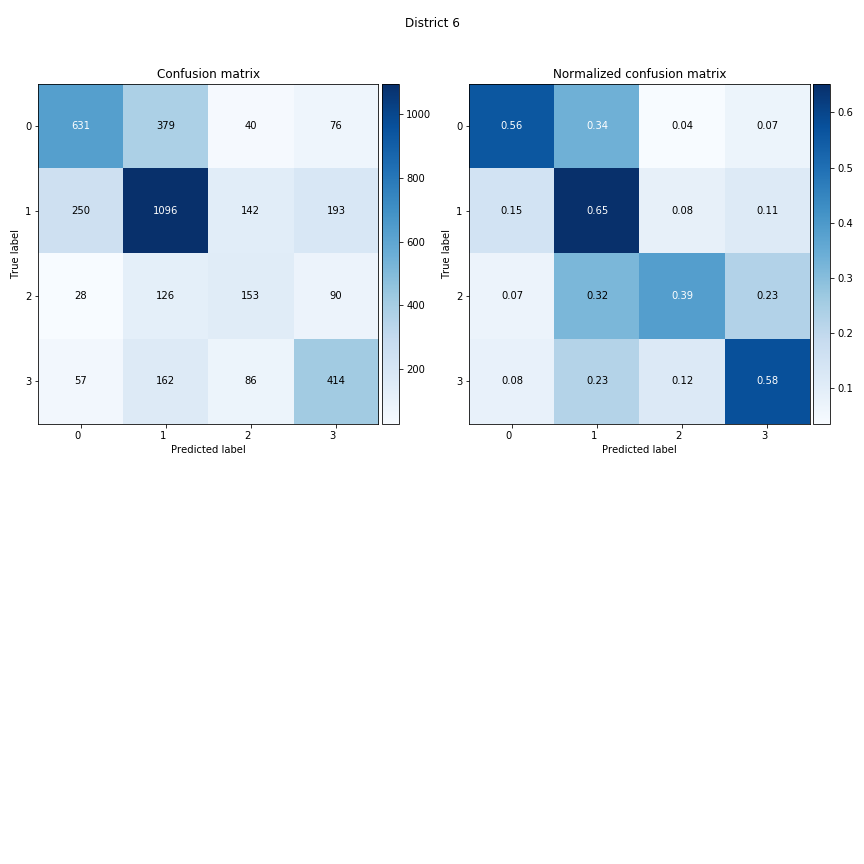
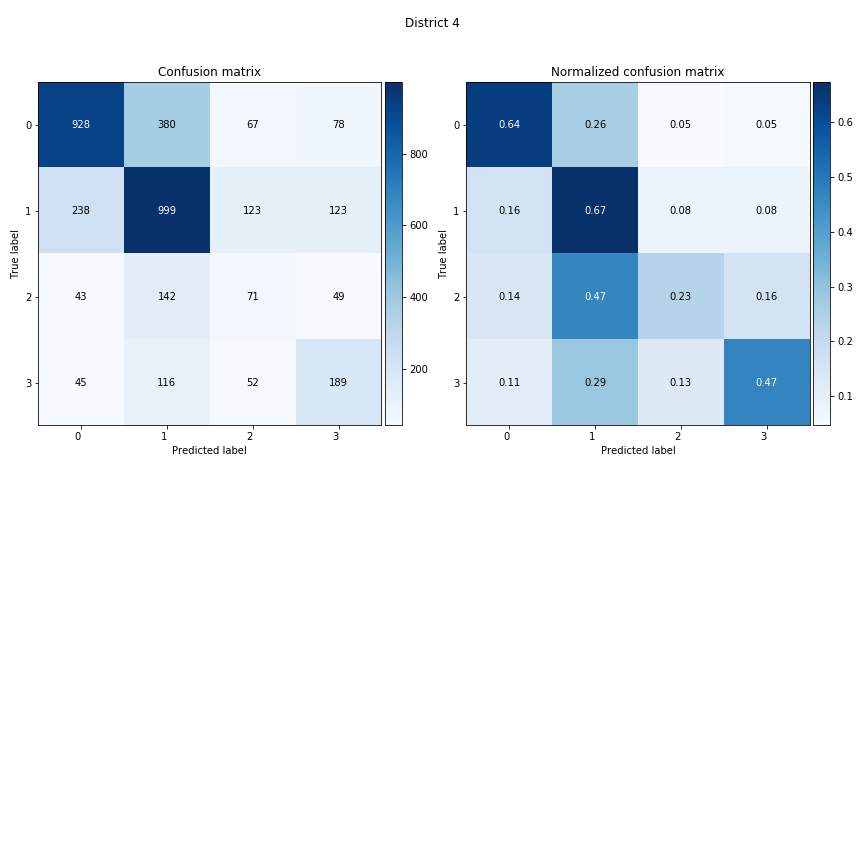
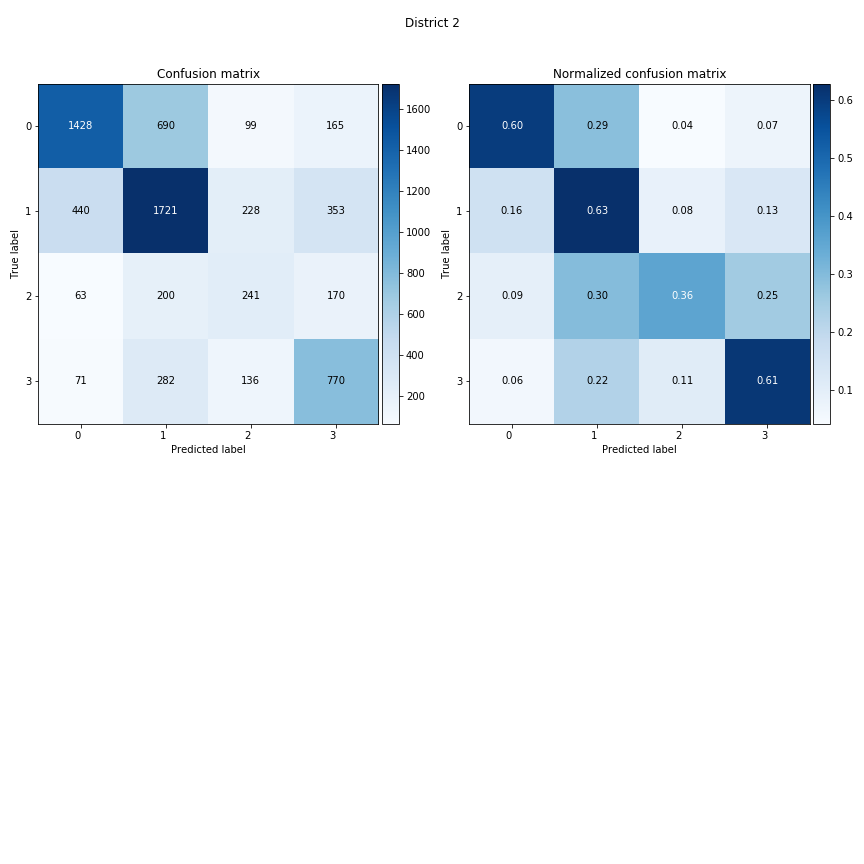
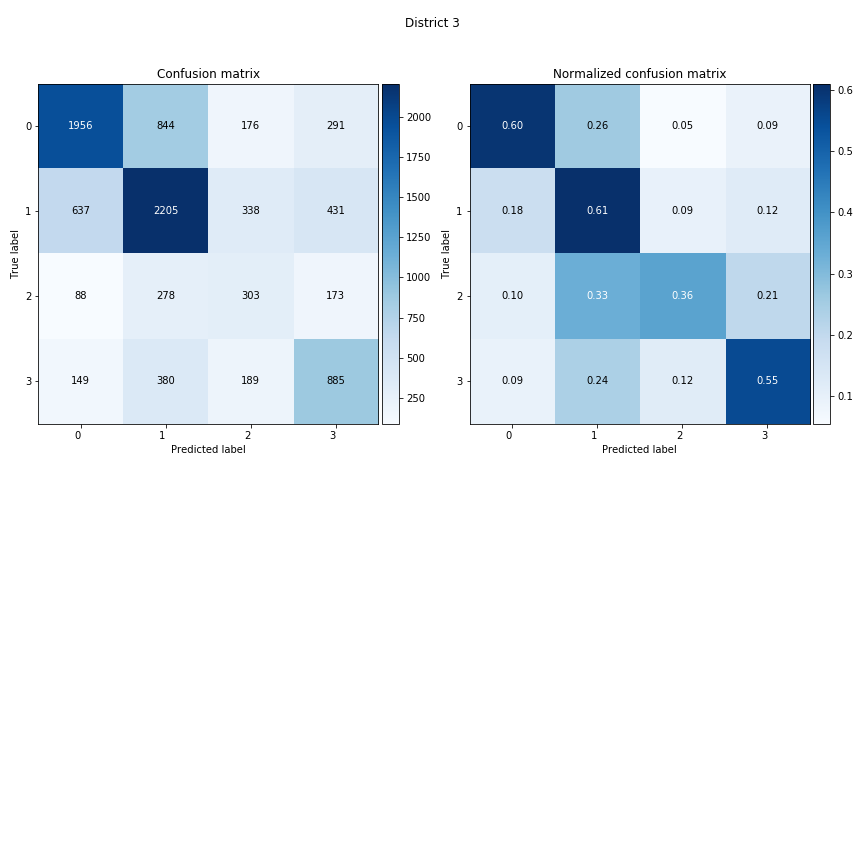
Website



Email



**Figure 7: Confusion matrices for requests categorized by method of communication**



District 1

District 2

District 3

District 4

District 5

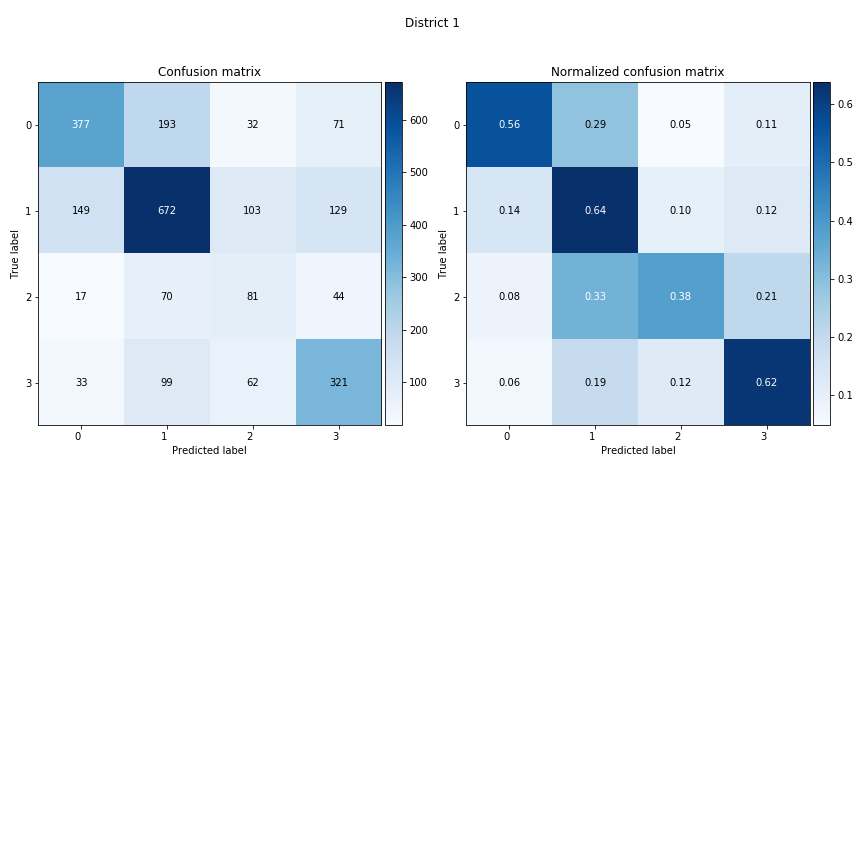
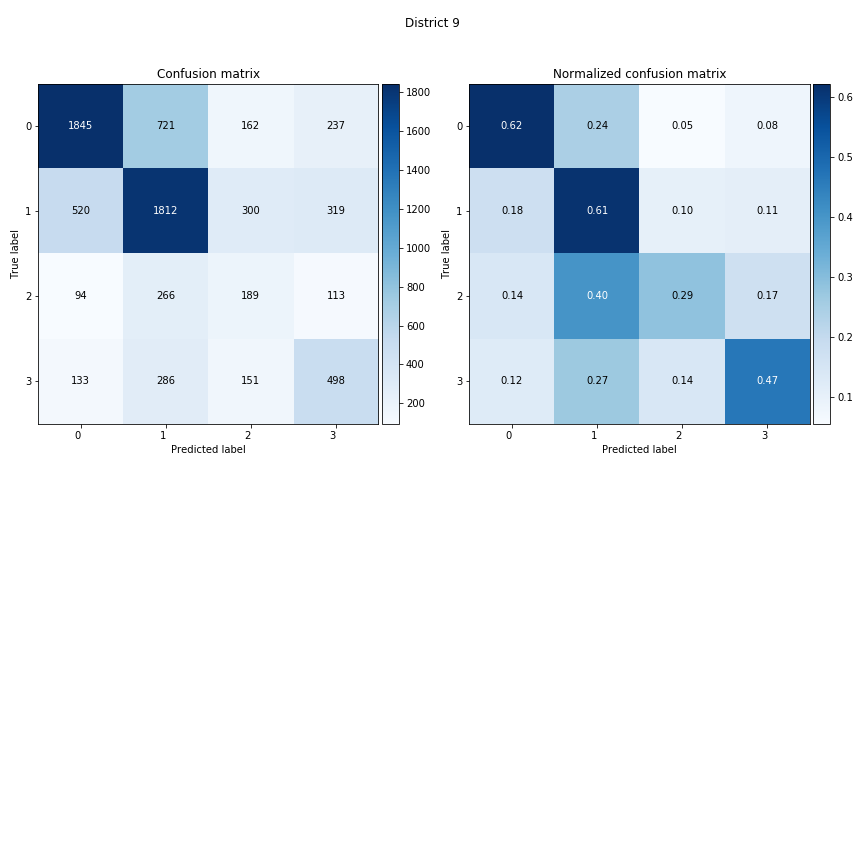
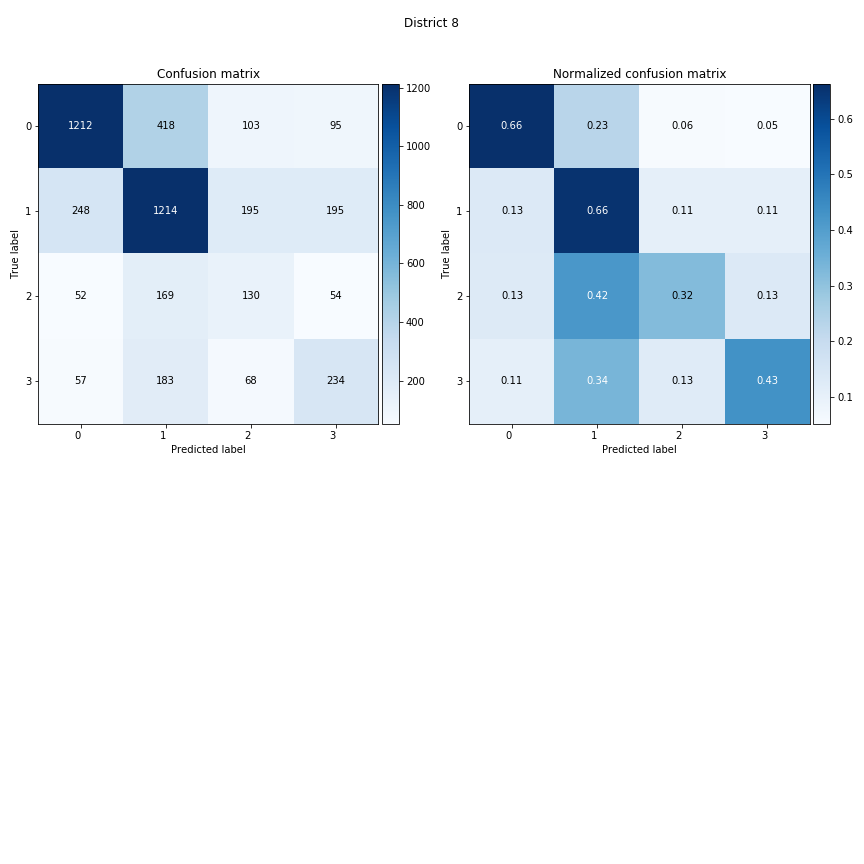
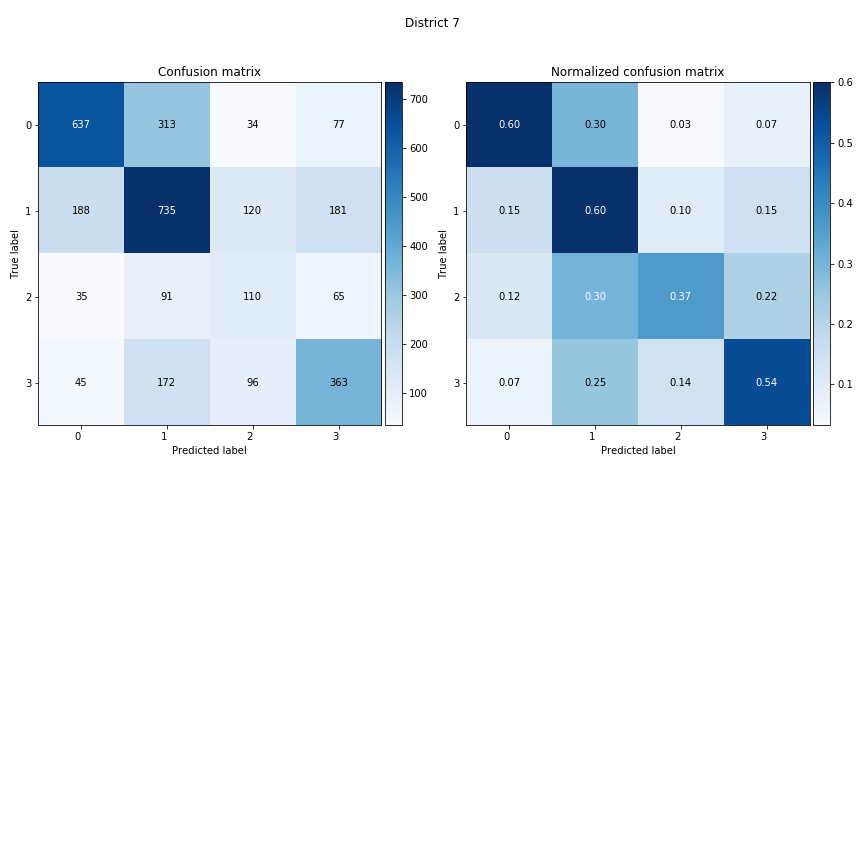
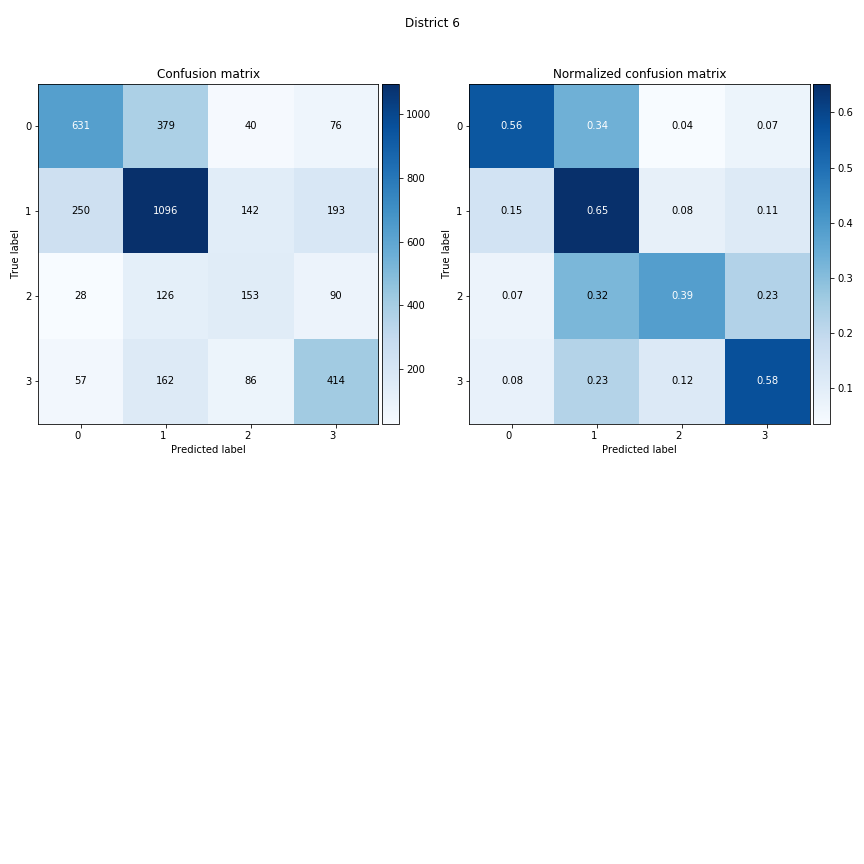
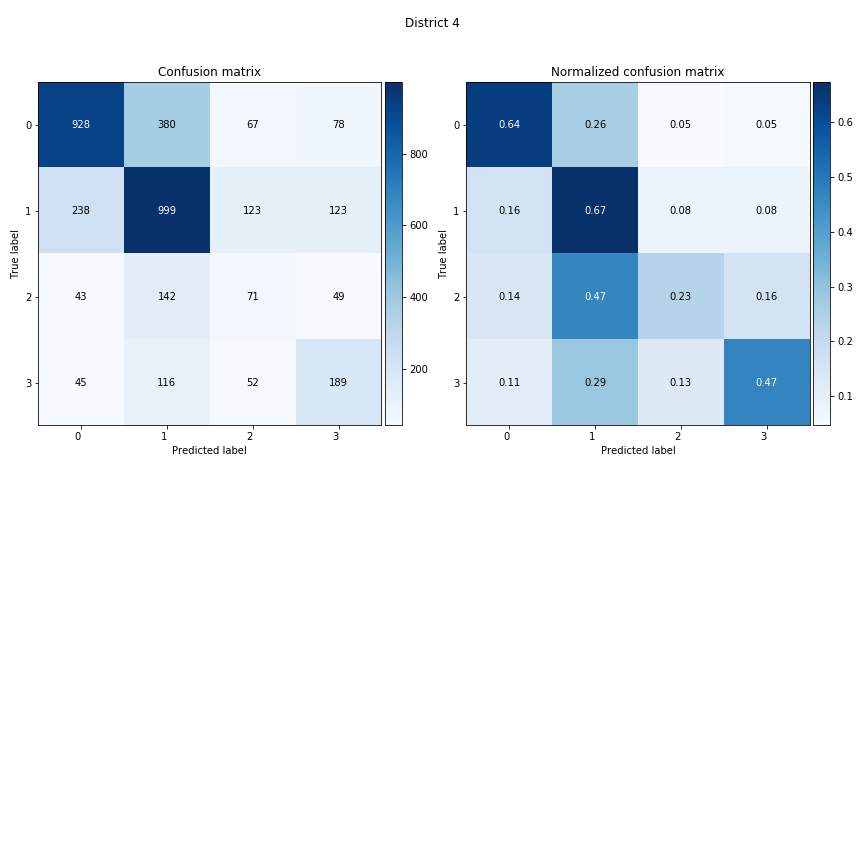
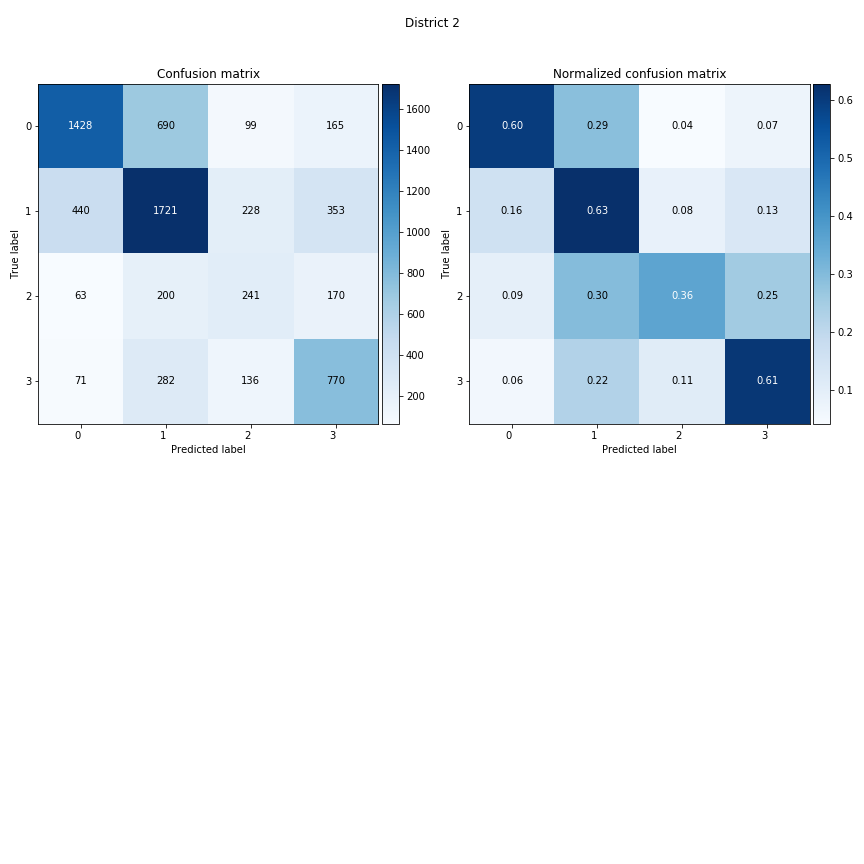
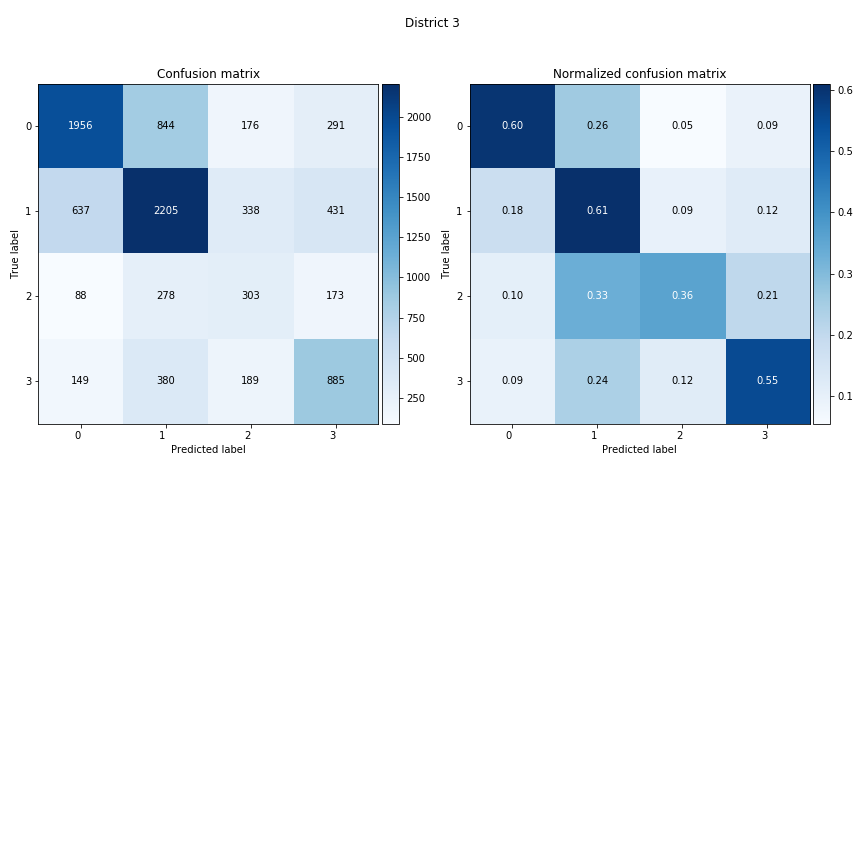
District 6

District 7

District 8

District 9

**Figure 8:** Confusion matrices for each City district (raw count). The main takeaway from this plot (and of Figure 4, next page) is the similarity in performance across districts.



District 1

District 2

District 3

District 4

District 5

District 6

District 7

District 8

District 9

**Figure 9:** Confusion matrices for each City district (normalized across true labels). As in Figure 3 (previous page) model performance is similar across districts.

**Conclusion**

The information available in the Get It Done San Diego database provides a wealth of useful information about the City’s needs and how the City responds to requests for service. The City of San Diego appears to have been successful at transitioning San Diego residents to this new method of communicating with city departments (the Get It Done San Diego mobile app and associated website), as seen by the steady increase over time in requests submitted both over the mobile app and the City website. There are a definite demographic-based differences in communication. Denser districts with less-wealthy residents are more likely to use the mobile app, neighborhoods with wealthier residents are more likely to use the City website, and less dense neighborhoods appear to prefer to call the City directly.

Overall, the mobile app appears to be an effective means of communicating with the City. Requests made via the app have the highest resolution and timely-closure rate, with a mean response time of 13.5 days. Requests made over the phone, in contrast, have the quickest response time (11.6 days) but the lowest resolution and timely-closure rates.

The City appears to be successfully distributing resources across the various neighborhoods, in that none of the demographic measures included here appear to strongly predict request closure rate. Timely closure rate was somewhat predicted by household income and population density, with less wealthy and denser neighborhoods having a higher closure rate. Because household income and population density are highly correlated, it is plausible that the City places a higher priority on closing requests in denser neighborhoods (and income is not the real predictor).

An attempt to create a model that could predict how long it takes to close any given request submitted to the City was moderately successful, but was able to provide only coarse estimates of the amount of time it would take to close a request, and model accuracy was inconsistent across service categories. Despite those weaknesses, the model could possibly be useful to provide a rough estimate of expected time-to-close when residents submit requests for service. The model may be more useful internally, in that it could be incorporated into the City’s tracking software and could flag requests that are taking much longer to resolve than the predicted time-to-close.

Two created features (load-per-service and load-per-service-zip) had the highest importances in the random forest classification model. It is not clear why those features are the most important to the model, but is an indication that additional feature engineering may improve model performance beyond what was accomplished here.

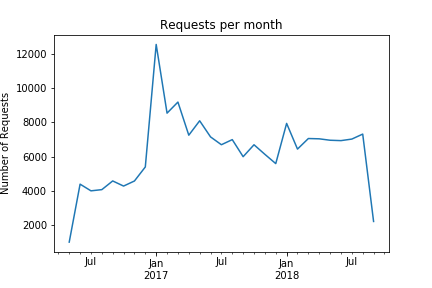
Finally, the model could be used to highlight underlying structural relationships in the data that are not otherwise immediately apparent. For example, it can be used to show which service categories have more predictable closure-times than others. It can also be used to show that resources do indeed seem to be distributed equitably across the City as seen by the nearly-identical model performance across districts. Other questions could be explored using this model if desired, and code could be developed to support this kind of exploratory analysis.

*You can reach me with questions or comments at mwesterfi@gmail.com*

**Appendix A: An analysis of duplicate requests in the Get-It-Done dataset**

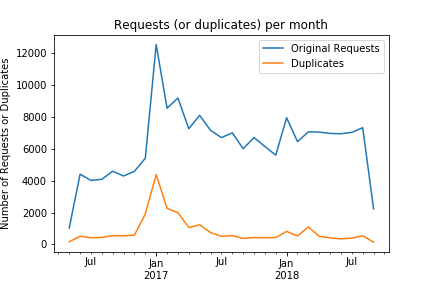
This analysis is motivated by what appears to be an aberration in the data. When looking at features such as total number of requests or how long it takes to close requests over time, there is a large peak at around January 2017, out of proportion to the rest of the data. I hypothesized that duplicate requests--service requests that were submitted multiple times--may have been a factor. In earlier versions of the dataset, several columns (parent\_case, duplicate\_verified, override\_duplicate) were devoted to tracking duplicates; hence this analysis will use a dataset downloaded on 9/11/2018 (as opposed to one that includes requests up to the current date).

If we look at the number of requests submitted over time, there is a surge starting in approximately 2/2017 and lasting until approximately 4/2017 (see Fig. 1)



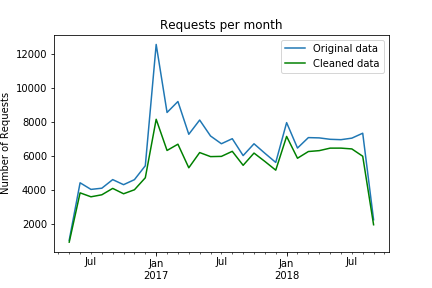
**Figure 1.** The spike in monthly number of requests in Jan. 2017

In Figure 2 below, the original monthly number of requests is plotted in blue, and the monthly number of duplicate requests is plotted in orange. It appears that the surge in the total requests is possibly influenced by the duplicate requests.



**Figure 2.** The spike in requests may be due to an underlying spike in duplicate requests

Once duplicate requests are removed from the dataset, we can see that the monthly number of requests is more consistent across time (Figure 3 below). There is still a surge from 1/2017 to 3/2017, which may accurately reflect a surge in number of requests during that time period, or perhaps implies that there are additional duplicate requests that were not successfully flagged in the dataset.

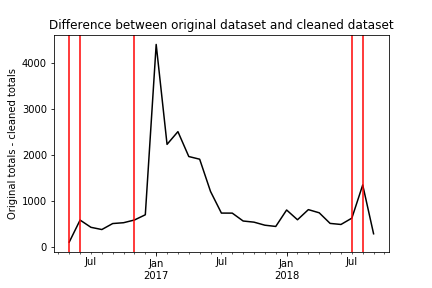


**Figure 3.** Removing duplicate requests results in more stable monthly request totals

*What's going on here?*

The plot below shows the difference between the monthly number of requests in the original data and in the 'cleaned' (duplicates removed) data. Numbers closer to zero mean that there is not too much discrepancy between the two datasets; the larger the value, the larger the impact of duplicate request submissions. Notice the sudden onset in duplicate requests starting 1/2017, which appears to taper off only gradually, appearing to return to baseline in 7/2017. This very abrupt increase in duplicates feels more like an artifact in the data than an accurate representation the number of service requests made.

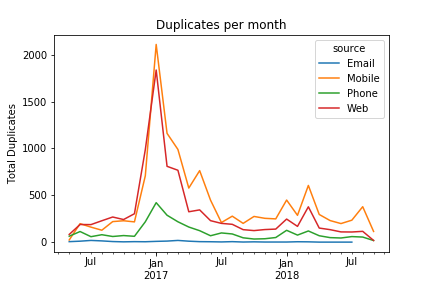
I suspected that perhaps this might correspond to an update to the app software. The red lines on the plot show the release dates of updates to the mobile app. I only have version history information from the iOS version of the app; Android version history is not available, although the app developers told me that the Android updates tracked the iOS updates fairly closely. Thus far, there is no clear correlation with app updates, although perhaps digging deeper into the data might reveal more (see below).



**Figure 4**. Red vertical lines indicate app updates

In order to examine whether the increase in duplicate requests were spurious due to an app update, I grouped the requests by the source of the request. The City tracks how the request was submitted; in order of frequency, the top sources are 1) the Get It Done mobile app, 2) the City of San Diego website, 3) a phone call to a City department, and 4) email to a City department. The plot below shows the monthly number of duplicate requests grouped by the source of the request.

Duplicate requests spiked in all groups except for email submissions; while the increase in duplicates in the phone submission group doesn't look as dramatic as the increases in mobile app or web submissions, that is more due to the low number of phone submissions overall. This makes it seem unlikely that an app update is at fault.



**Figure 5.** Duplicate requests were not limited to mobile app submissions

**Conclusion and recommendation for data processing**

A large number of duplicate service requests in the Get It Done dataset appear to be spurious, and may not contribute useful information in an analysis of the City's ability to provide services to its residents. If these duplicate requests were evenly distributed throughout the data, they might not be harmful, but their concentration during the first half of 2017 most likely will skew any analyses (especially time-based ones) in an unpredictable manner.

The City has revised the content of this dataset several times over the past few months. As of January 2019, all information about duplicates has been removed, and this change is retroactive (affects all request records back to 5/2016). I have verified that in the 01/2019 version of the dataset, requests that were previously coded as duplicates are still included in the dataset, but are no longer tagged as duplicates. Thus there are two options for subsequent analyses:

1. Only analyze data from 5/2016 through 11/17/2018 (the most current download I have that still contains duplicate flags)

2. Drop duplicates from records through 11/17/2018, but append records from 11/18/2018 up to current date. There will be (unidentifiable) duplicate requests in the data from 11/18/2018 on; this option will assume that there are not an excessive number of duplicates.

Each subsequent analysis of this dataset will specify which option was used.

1. https://www.sandiego.gov/sites/default/files/2015-cosd-resident-survey.pdf [↑](#footnote-ref-1)
2. https://data.sandiego.gov/datasets/get-it-done-311/ [↑](#footnote-ref-2)
3. https://www.sdvote.com/content/dam/rov/en/maps/SDCityCouncilMap.jpg [↑](#footnote-ref-3)
4. https://pypi.org/project/uszipcode [↑](#footnote-ref-4)
5. http://datasurfer.sandag.org/ [↑](#footnote-ref-5)
6. https://blog.splitwise.com/2014/01/06/free-us-population-density-and-unemployment-rate-by-zip-code/ [↑](#footnote-ref-6)