San Francisco Crime Classification

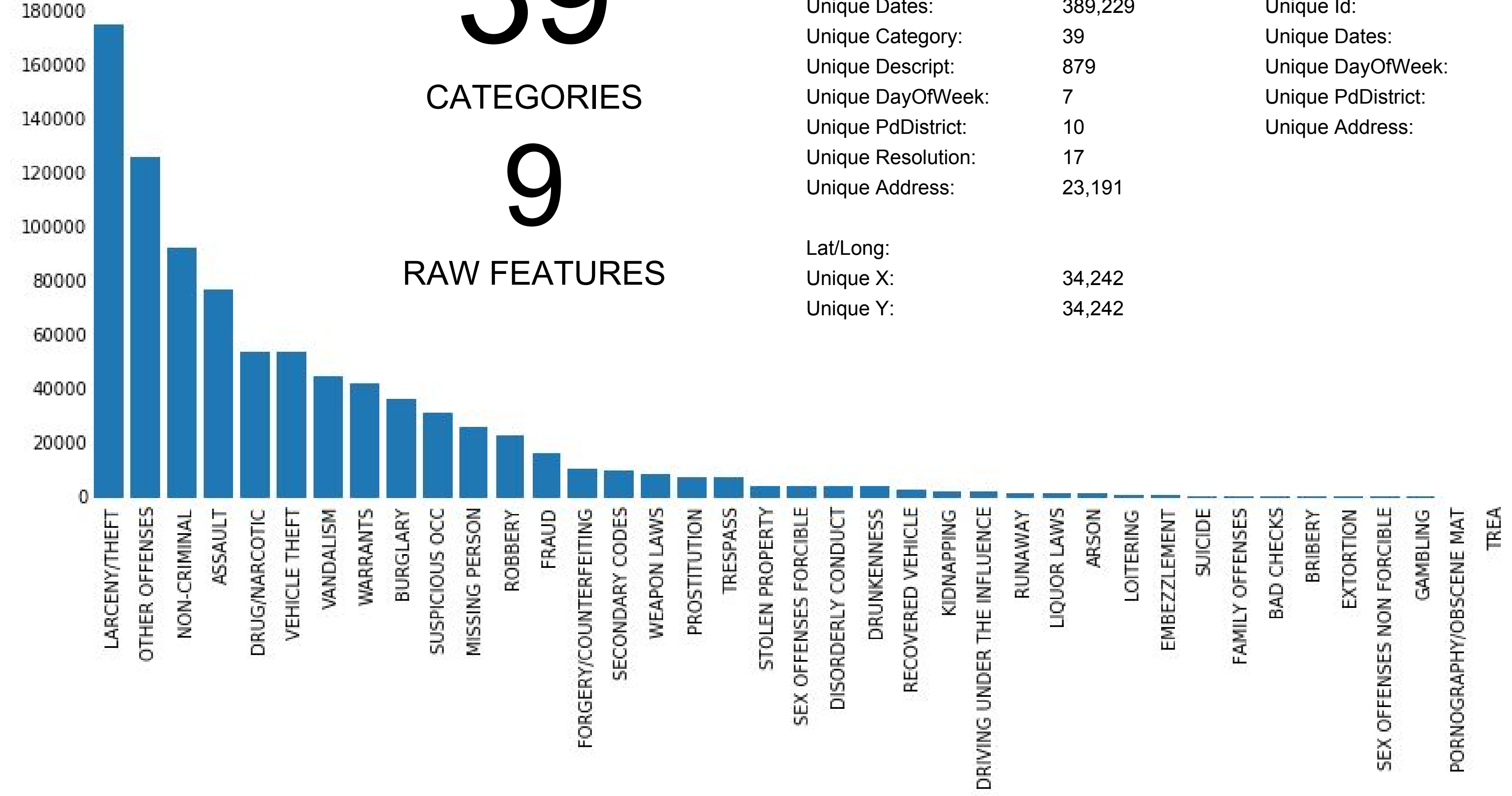
Kaggle competition
https://www.kaggle.com/c/sf-crime

Kyle Hamilton, Amitabha Karmakar, Jackson Lane

W207 - Machine Learning - Spring 2016
Daniel Percival, UC Berkeley

Data Exploration

https://public.tableau.com/views/SanFranCrimeStory/Story1?:embed=y&:display_count=yes&:showTabs=y



39

CATEGORIES

9

RAW FEATURES

Raw Training data features

Unique Dates:	389,229
Unique Category:	39
Unique Descript:	879
Unique DayOfWeek:	7
Unique PdDistrict:	10
Unique Resolution:	17
Unique Address:	23,191

Lat/Long:	
Unique X:	34,242
Unique Y:	34,242

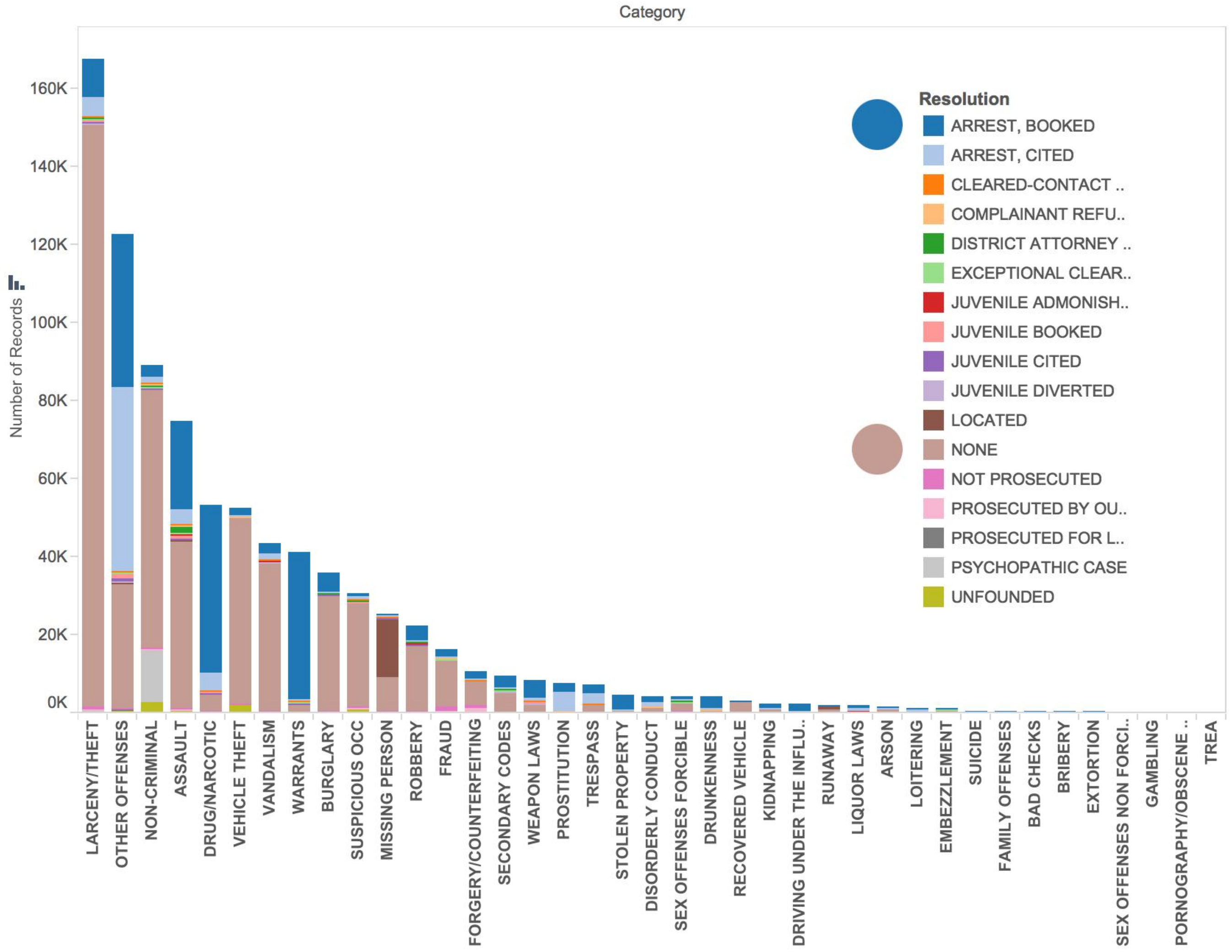
Raw Test data features

Unique Id:	884,262
Unique Dates:	392,173
Unique DayOfWeek:	7
Unique PdDistrict:	10
Unique Address:	23,184

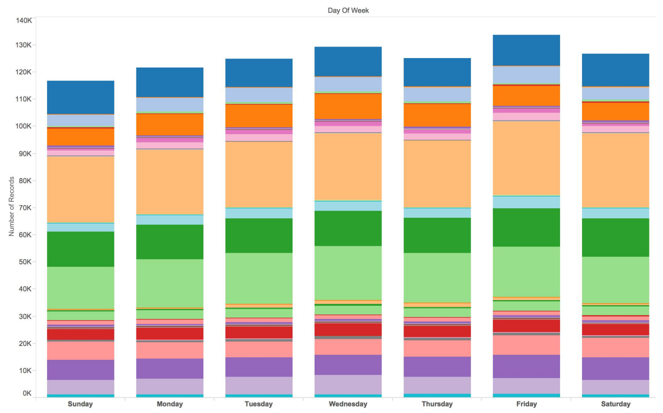
The data is not balanced: the majority of the observations fall into a handful of categories.

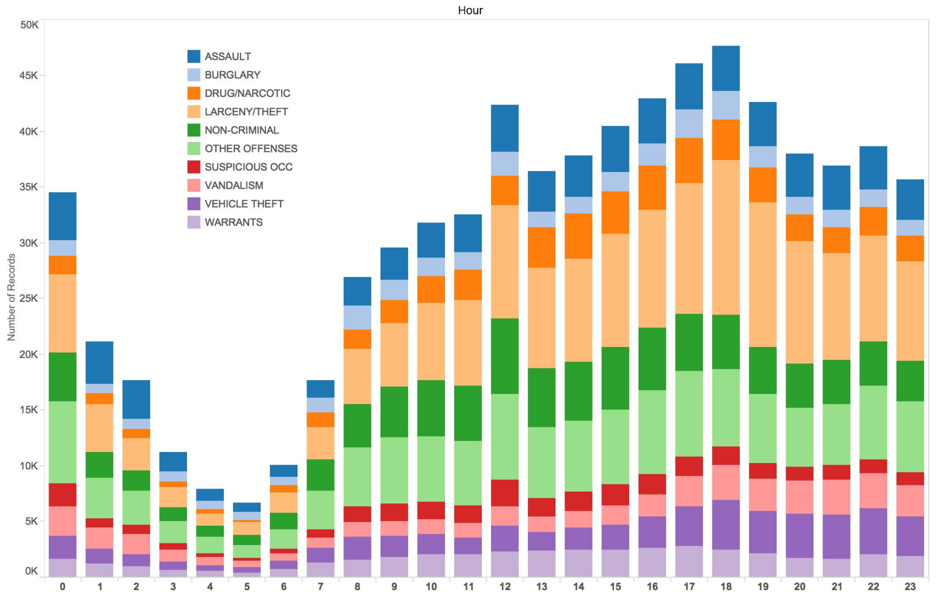
The least represented category - TREA - contains a mere three observations.

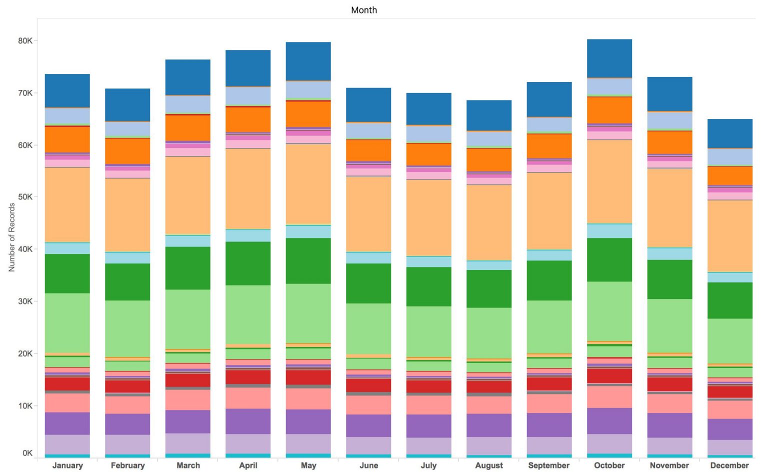
The training data contains a column for resolution. The test set does not contain this column. This is unfortunate, as it seems the most obvious differences are in the resolution ratios.



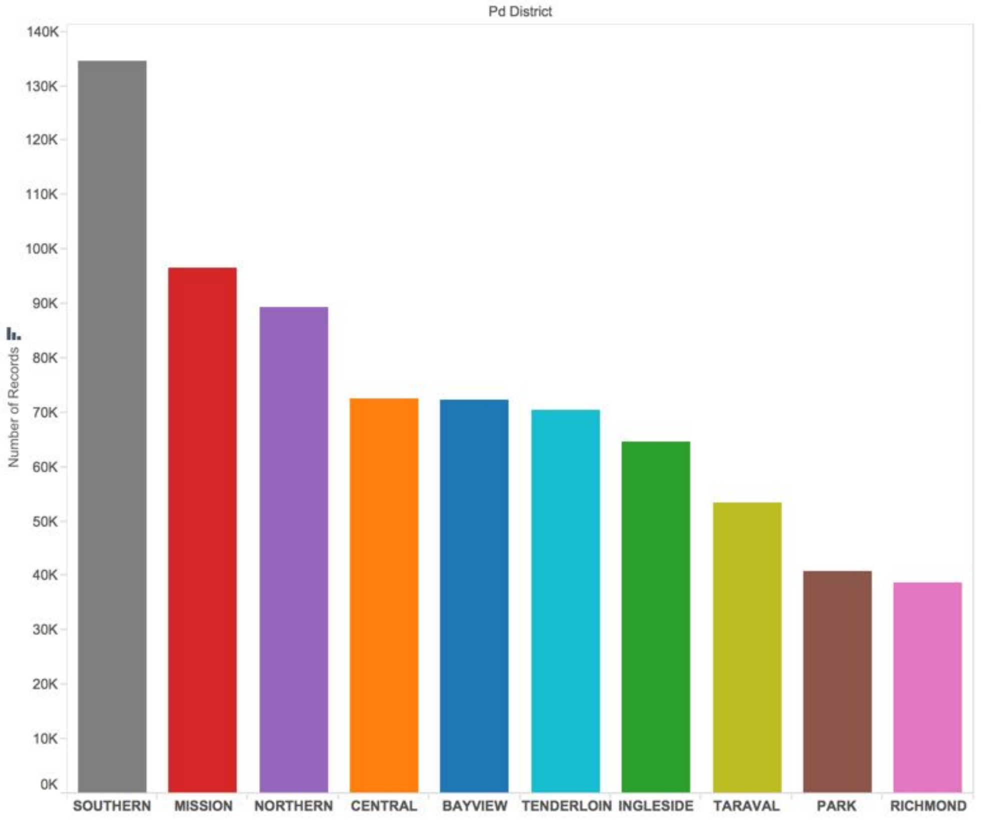
Criminal activity by **hour**, **day**, and **month**, colored by category (top 10 are listed). A similar pattern is seen for most categories.

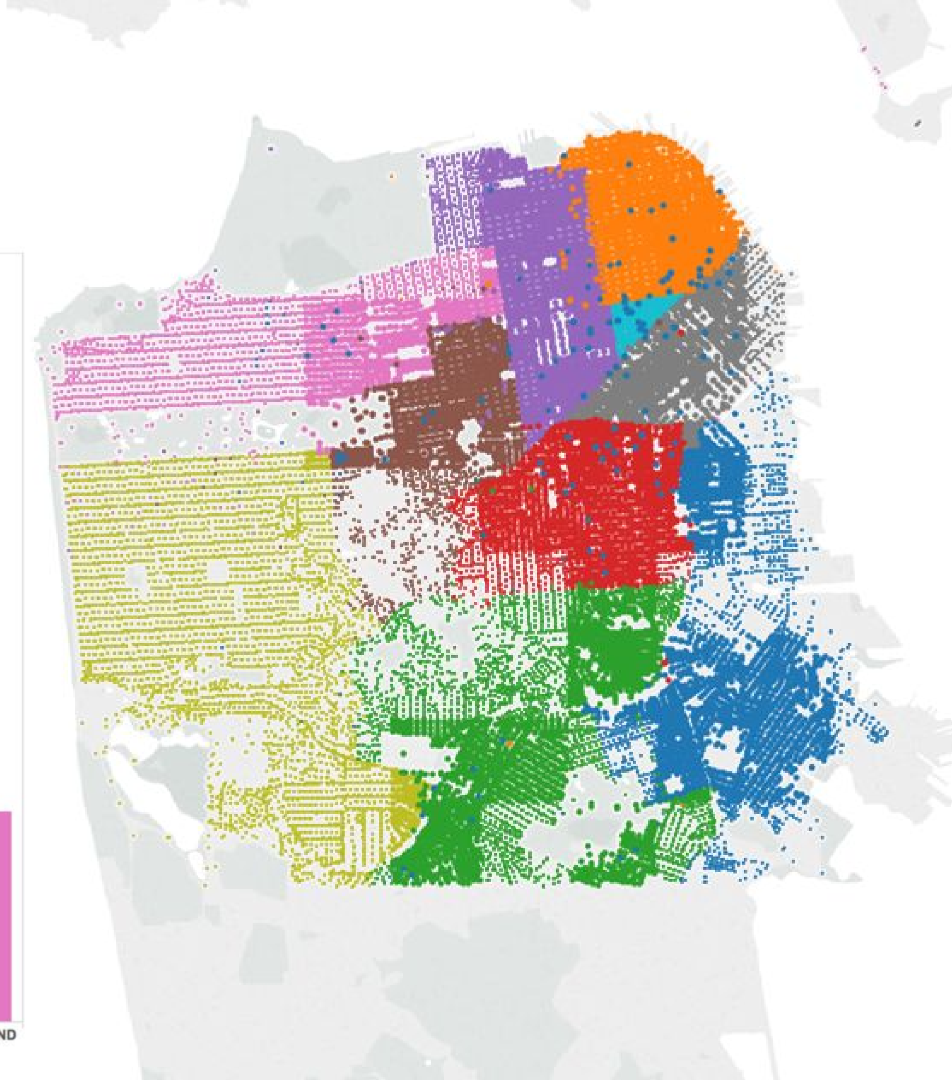






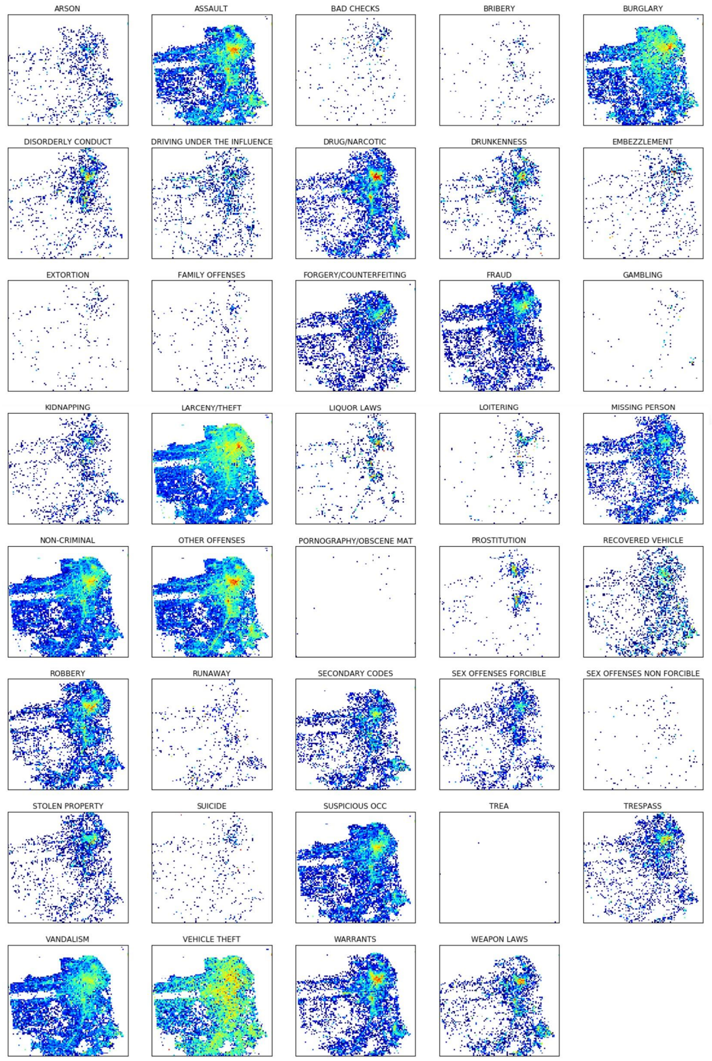
Most activity is concentrated in the north-eastern section of the city - in particular the "Southern" police district (a misnomer)





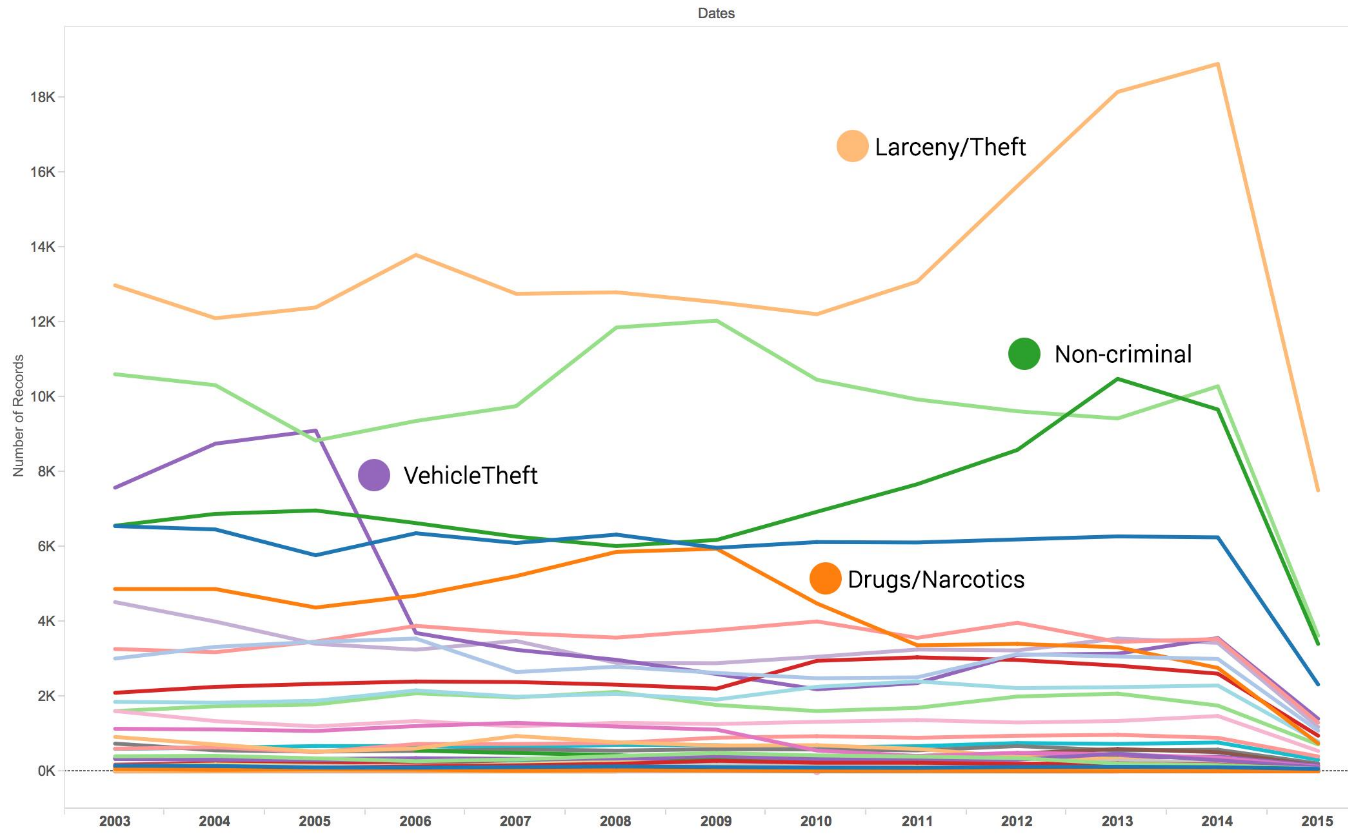
For the sake of completeness, we include heatmaps for each category.

There are no obvious geographic patterns distinguishing the categories beyond the number of occurrences.



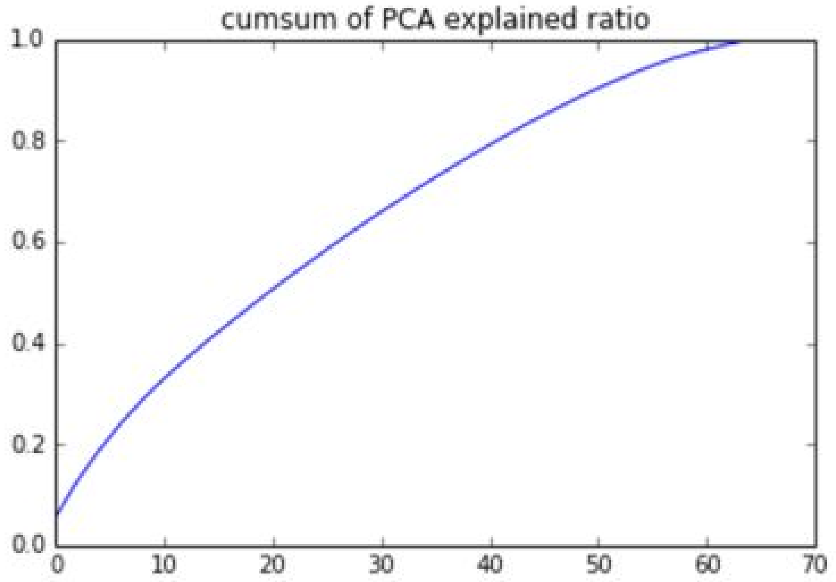
Activity over time.

At first glance, there appears to be a sudden drop in **2015**. This is due to having only five months of data for that year.



Feature Engineering

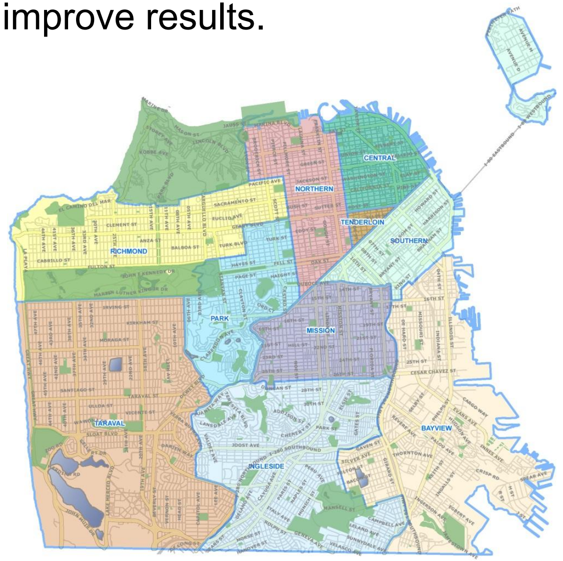
- Engineered 70 features
  - Log-odds
  - Onehot
  - Parsed date fields
- Log-odds: odds of crime category given specified fields
  - Eg. Odds that a crime
- Varied log-odds fields to train models in ensemble
  - Address
  - Time, District, Address
  - Street names, Hour of day
- PCA shows not all features were necessary

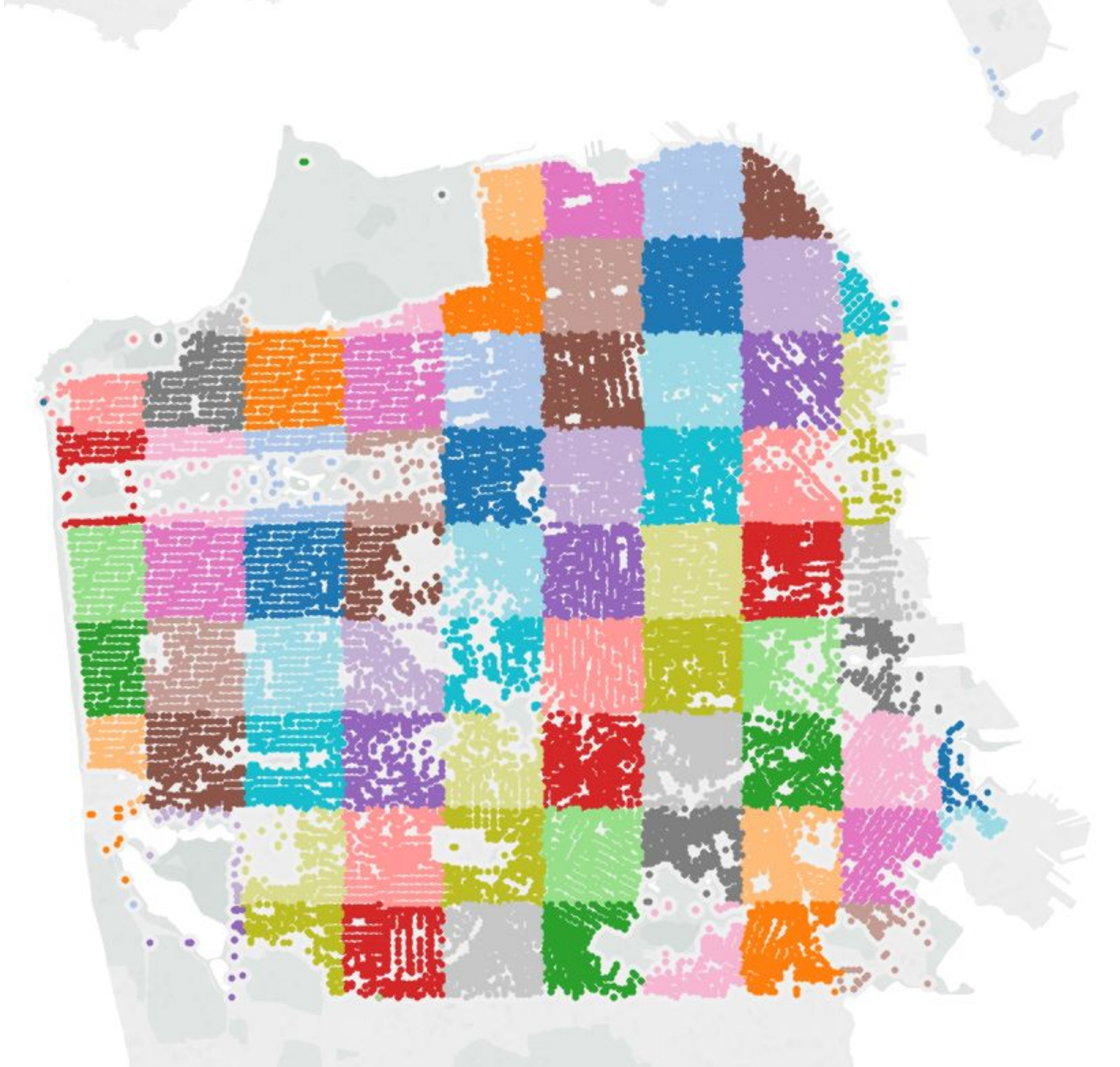


The data contains a column for the police district, and there are 10 police districts in San Francisco.

We create more granular geographic regions by creating an N x N grid from the Lat/Long values. We try different values of N from 10 (on the right) to 100.

Surprisingly, this granularity did not improve results.

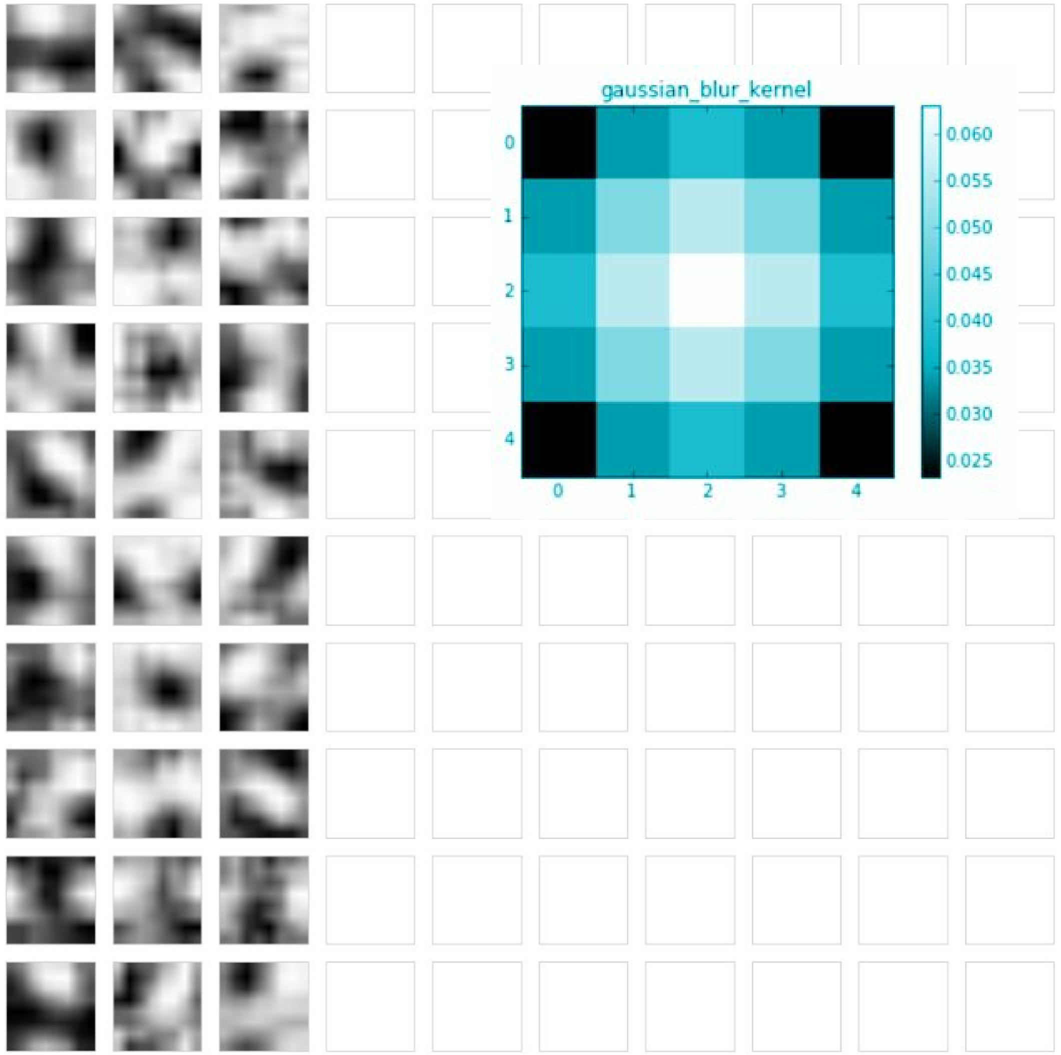




Given the engineered features, we run **PCA** to reduce the feature set to 64. We reshape each vector into an 8x8 matrix, and apply 5x5 gaussian blur filter.

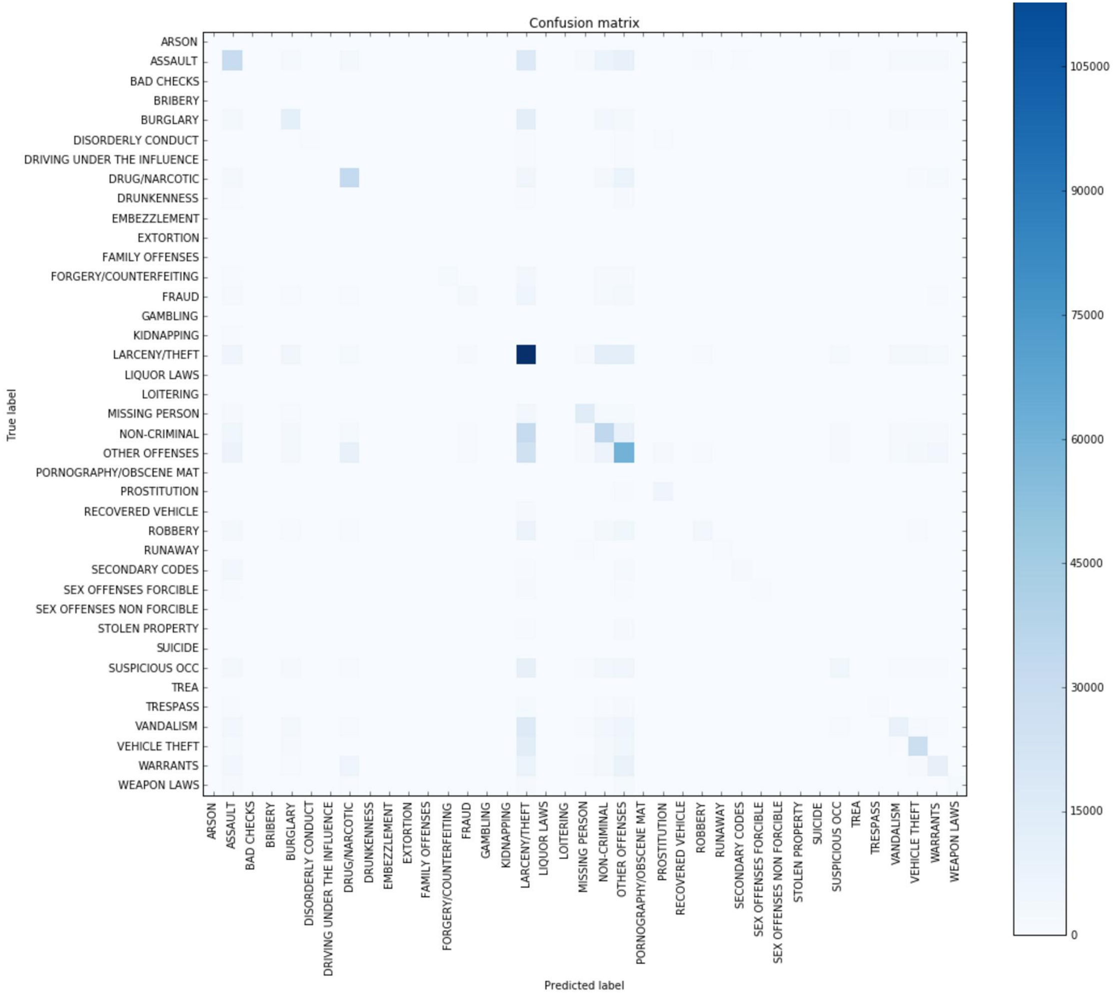
In the figure on the right, each row contains 3 examples of a given category (only 10 categories are represented).

There are no visible patterns, and it is not surprising that this feature engineering effort did not improve results.



Error Analysis

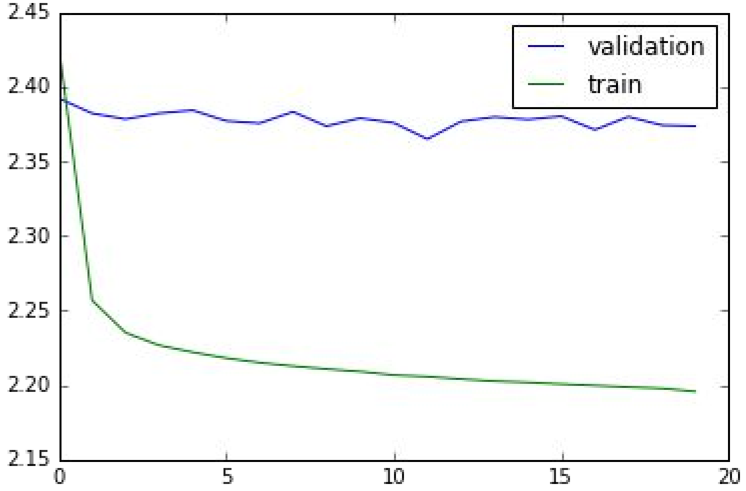
- Most confusion with
  - Larceny/Theft
  - Other Offenses
  - Assault
  - Non-Criminal
- Most confused categories tended to be most populous
- Next step: Put more weight on IsIntersection
  - Other Offenses more likely to occur at intersections than other categories are



Overfitting

- Overfit model by
  - Increasing hidden nodes and epochs
  - Adding more fields to log odds
- We were not able to detect due to coding error
  - Result: Log loss score on Kaggle much higher than what we predicted
- Useful in ensemble averages model
  - Varied feature engineering and initialization parameters
-

Example of overfitting
Neural Network -
20 Epochs, 1024 Hidden Nodes



Results

Best Ensemble of NNs Kaggle Score:

2.21925, Rank: **15 / 1842**

Best Single Neural Network Kaggle Score:

2.23206

Best Logistic Regression Kaggle Score:

2.43958

Best Random Forest (not submitted to kaggle - using dev-data)

2.4720487101

Best Multinomial Naive Bayes Kaggle Score:

2.57048

Models

Adding the best **Logistic Regression**, **Multinomial Naive Bayes**, and **Random Forest** models to the ensemble did not improve results.

We did not pursue tuning these models, instead we focus on Neural Networks.

We use the Keras library to stack the layers of our network. We start with code provided by user papadopc, and experiment with various combinations of activation functions, regularization, objective functions, and optimization functions. The best results are obtained from the original stack. On the other hand, we found that the optimal number of epochs to be ~50, and the optimal number of hidden nodes to be ~512. Values higher than those proved prone to overfitting.

Dense : We start with “Just your regular fully connected NN layer”.

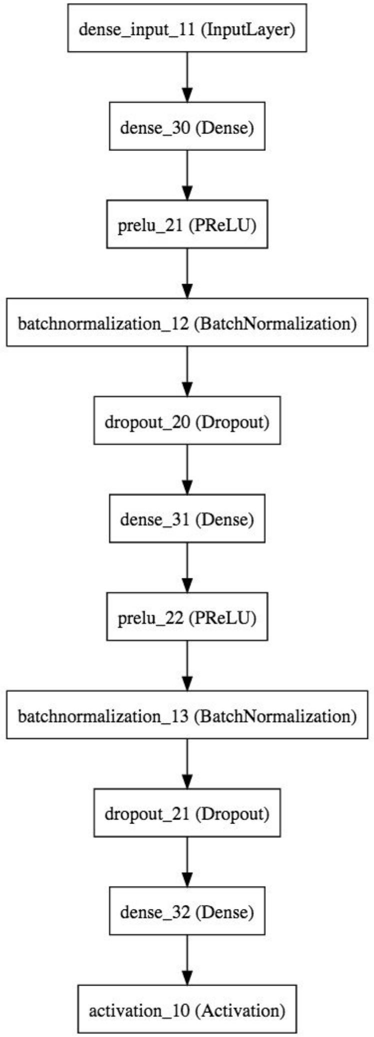
We try out **Maxout Dense**. According to the literature, using a maxout in conjunction with dropout, should improve performance. However, this was not the case in our model and produced NaNs when evaluating the log-loss.

We experiment with several activation functions, including: *softplus, relu, tanh, sigmoid, hard_sigmoid, and linear.* **Softmax** gives us best results - it is the recommended activation layer for problems with many classes.

We keep the Parametric Rectified Linear Unit (**PReLU**) layer. According to the literature, PReLU improves model fitting with nearly zero extra computational cost and little overfitting risk.

Regularization:
We include **Dropout** and Gaussian Noise to reduce overfitting, as well as **Batch Normalization** to address covariance shift between batches. Batch normalization also contributes to regularization.

Objective (loss) function: a multiclass logloss
Optimization function : Adam


dense_input_11 (InputLayer)
dense_30 (Dense)
prelu_21 (PReLU)
batchnormalization_12 (BatchNormalization)
dropout_20 (Dropout)
dense_31 (Dense)
prelu_22 (PReLU)
batchnormalization_13 (BatchNormalization)
dropout_21 (Dropout)
dense_32 (Dense)
activation_10 (Activation)

Ensembles

- **Different models** - combinations of similarly performing NNs work well.
- **Same model, different initializations**. Use cross-validation to determine the best hyperparameters, then train multiple models with the best set of hyperparameters but with different random initialization. The danger with this approach is that the variety is only due to initialization. http://cs231n.github.io/neural-networks-3/

Conclusions

NN’s are hard.

Feature Engineering is hard.

Lessons learned:

NNs take a long time to train

It’s easy to forget what you’re doing, so.. should save results at various stages for future reference in an organized manner. Document as you go! Due to lack of experience, we did not follow this strategy. Next time we will design a better pipeline that includes this.

The dataset is limited. The kaggle rules do not permit augmenting with external data. In real life there is an opportunity to merge with other data sets, zip codes, demographics, weather data. One API in particular returns nearby places - such as “liquor store” with a specified radius. We hypothesize that this level of information would be very valuable.

References

http://ieeexplore.ieee.org/xpls/icp.jsp?arnumber=6033567

http://cs231n.github.io/neural-networks-2/

http://arxiv.org/pdf/1310.4546.pdf (heirarchical softmax)

http://arxiv.org/pdf/1302.4389.pdf (Maxout Dense)

http://arxiv.org/abs/1502.03167 (Batch Normalization)

http://arxiv.org/abs/1412.6980v8 (Adam optimization function)

http://keras.io/

http://mlwave.com/kaggle-ensembling-guide/ (ensemble guide)