IML Hackathon 2019: Crimes in Chicago

Data Processing

Data exploration

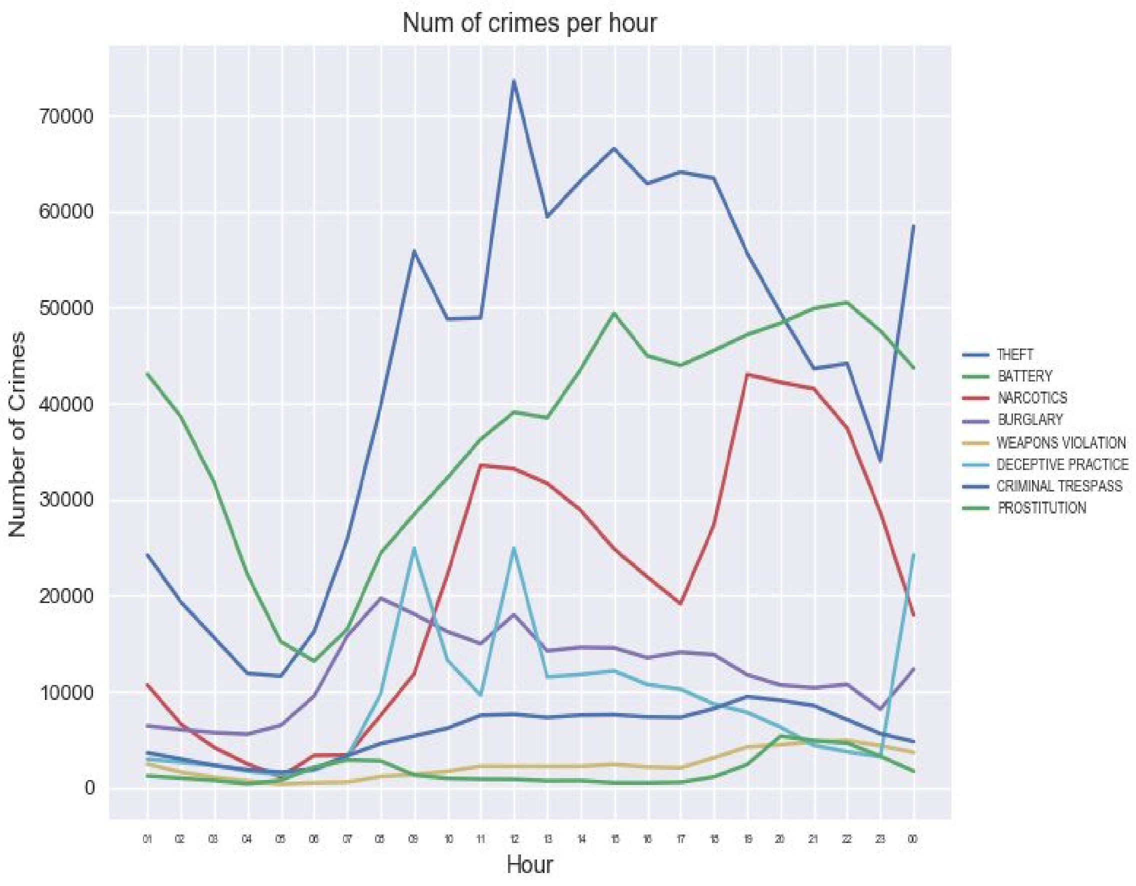
The input is 20 columns: 2 bool, 9 float, 4 int, 5 string. The floats aren't actually floats, but really integers. We've downcasted all numerical data types, and categorized all string (object) columns in order to reduce memory footprint and improve efficiency.

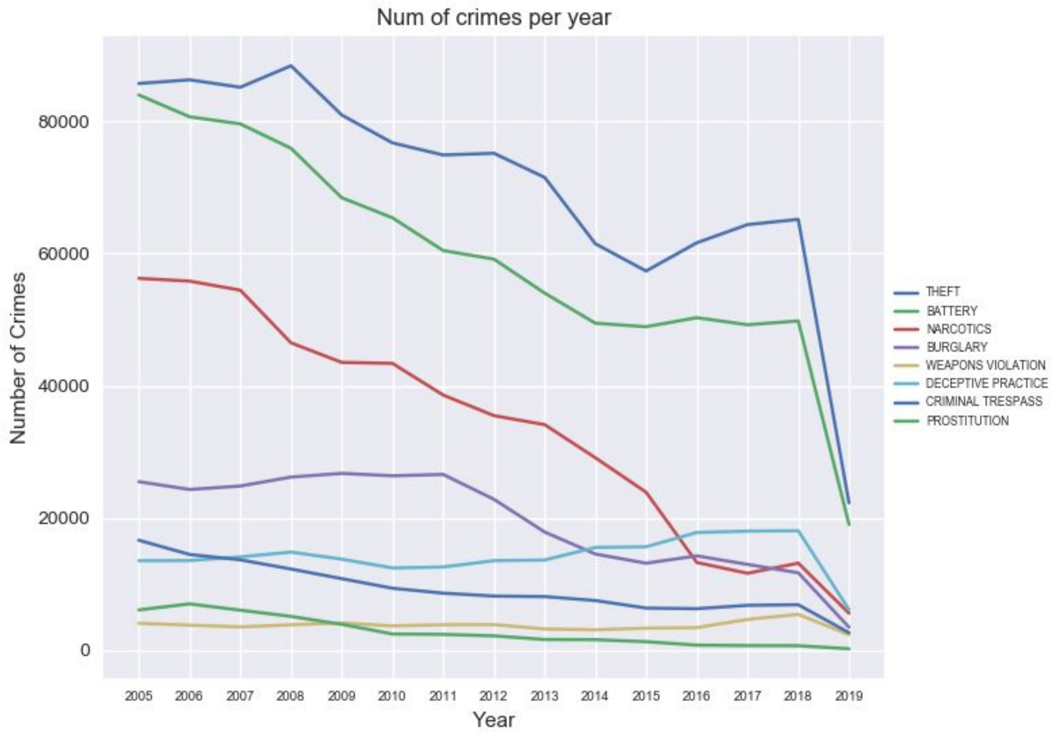
Data Cleaning

We've removed clearly redundant columns ('Update On', 'Unnamed', 'ID'). We checked number of nulls per feature, and found no significant lack of data. The feature with most NaN was only missing data in 0.15% of the rows.

Feature Enginerring

The most surprising experience we had with the data is that removing input features improved performance significantly. This is an example of the Bias-Variance tradeoff - the simpler we made our model, the better it could generalize. Extracting the hour of the crime, the month and the year improved our models. Using the day of the month a crime occured was significantly less useful for classification.





Models

1. <u>KNN classifier</u>: After doing hyperparameter tuning (ranging from 1 to 100) using cross-validation, we've found k=10 to be good candidate. We've achieved 48% accuracy with this classifier.
2. <u>Decision Tree Classifier</u>: A benchline to see how much can the Random Forest will improve. This classifier had about 58% accuracy.
3. <u>Random Forest</u>: The winning model. Again, using hyperparameter tuning, we've settled on 70 trees, max_depth of 50, min_samples_split=30, and min_samples_leaf=25. This model was accurate 70% of the time.
4. <u>Multi-Layer Perceptron</u>: the biggest dissapointment. We've used an MLP with a single hidden layer of 40 nodes. This model achieved 60% accuracy.

Interesting Challenges

1. Testing different models was the easier part of the project. Cleaning the data and extracting features was the real challenge.
2. Dealing with NaN: We have to provide a prediction for all rows, even ones lacking data. We've fiddles around with interpolation and Multivarite Imputation, but simply replacing NaNs with the last valid value worked just as well. This is probably due to the large dataset we're given.
3. Categorizing posed several challenges: first, we needed to save the mapping we've applied in training time to test time. Moreover, we can't assume that our training data will contain all possible categories. Additionally, a well-thought categorizing method should be able to handle completely new categories in an adequate manner. Due to all this problems, we've tried running our codes without the categorical variables, and to our surprise, the classifiers' performance improved.