A Recommendation Engine to Aid in Identifying Crime Patterns

Alex Chohlas-Wood* **and E.S. Levine***

*New York City Police Department, New York, NY 10038

Police investigators are routinely asked to search for and identify groups of related crimes, known as patterns. Investigators have historically built patterns with a process that is manual, time-consuming, memory based, and liable to inefficiency. To improve this process, we developed a set of three supervised machine learning models, which we called *Patternizr*, to help identify related burglaries, robberies, and grand larcenies. Patternizr was trained on ten years of manually identified patterns. Problematic administrative boundaries and sensitive suspect attributes were hidden from the models. In tests on historical examples from New York City, the models perfectly rebuild approximately one-third of test patterns and at least partially rebuild approximately four-fifths of these test patterns. The models have been deployed to every uniformed member of the New York City Police Department through a custom software application, allowing investigators to prioritize crimes for review when building a pattern. They are used by a team of civilian crime analysts to discover new crime patterns and aid in making arrests.

law enforcement | pattern recognition | machine learning | decision support
Correspondence: ***alexcw@stanford.edu, evan.levine@nypd.org***

Introduction

Predictive policing can be defined as "the application of analytical techniques—particularly quantitative techniques—to identify likely targets for police intervention and prevent crime or solve past crime by making statistical predictions" (1). These techniques have come into increasingly common usage as the amount of data available to law enforcement has grown (2, 3). Law enforcement agencies have a tradition of using quantitative techniques to inform operations. Pioneering techniques in queuing (4, 5), in resource allocation via hotspots (6), and in organizational management (7) have been developed and implemented in police departments worldwide. Continuing this tradition, a wide variety of predictive policing applications are in use by law enforcement agencies today, including algorithms that forecast geographies at high risk of imminent crime (8, 9), predict recidivism to aid in sentencing (10), or identify individuals at risk for involvement in future gun violence (11). Advanced analytics have also been used in police management, such as optimizing the recruitment process for police applicants (12), tracking the locations of officers on patrol (13), and identifying officers in need of counseling or training (14).

This is an electronic reprint of the original article published by the INFORMS Journal on Applied Analytics, 2019. This reprint differs from the original in pagination and typographic detail.

One topic that has received less attention is the process of identifying crime patterns. Researchers have outlined potential approaches for using advanced analytics to aid in the identification of crime patterns by connecting similar crimes (15, 16), as analogous recommendation engines have been the subject of much attention in other domains (17); however, we know of no previous production deployment of these algorithms in a law enforcement environment. To aid New York City Police Department (NYPD) investigators in identifying groups of related crimes, we have designed and implemented a recommendation algorithm, which we call Patternizr.

In 2016, people in New York City reported approximately 13,000 burglaries, more than 15,000 robberies, and over 44,000 grand larcenies (18). Evidence from the legal system and criminological research indicates that many of these crimes were committed by serial offenders (19). A series of such crimes—committed by the same criminal—is known as a pattern. The crimes in a pattern may be nearby in space, similar in times and days of occurrence, and (or) alike in method (also known as the modus operandi, or M.O.). Once a pattern has been identified, it is possible for police to use evidence from related crimes to more easily identify and apprehend the perpetrator. Additionally, if police investigators have connected an arrest to multiple crimes as part of a pattern, they can close more investigations and prosecution can proceed with more accurate charges.

However, identifying patterns is not an easy task. Investigators at many police departments follow a manual and memory-based pattern identification process that has not changed much from the days of paper records (20). While reading through a crime report, known as a complaint, an investigator attempts to recall past crimes with similar characteristics. If two complaints seem to the investigator to be similar and could plausibly be committed by the same individual, the investigator links the two complaints into a pattern. Because investigators focus most of their limited time on recent complaints in their own precincts, patterns that occur across significant distances (particularly across multiple precincts) or over long periods are especially difficult to identify. Although modern technology has augmented this memory-based process with simple search engines (21), these searches are typically done using exact categorical matches, rather than searching for broader similarity across many disparate crime characteristics. Collectively, the NYPD still spends hundreds of thousands of hours each year searching for patterns using these methods.

Solution Design

Patternizr is a decision support tool and recommendation engine that consists of a set of machine learning models to make this search process more efficient and effective. The purpose of Patternizr is to take a "seed" complaint, chosen by the analyst, and compare this seed against the hundreds of thousands of recorded crimes in NYPD's crime database. With each comparison between a crime and the seed, Patternizr generates a "similarity score" that quantifies the likelihood that the pair of crimes are in a pattern together. Once all similarity scores have been generated, Patternizr ranks the pairs in descending order by similarity score and returns a list for the analyst to review. The complaints most likely to be in a pattern with the seed complaint appear at the top of the list. After reviewing these ranked results, the analyst can decide which complaints should be grouped together into a pattern.

The models that comprise Patternizr are supervised machine learning classifiers; that is, they are statistical models that learn from historical examples where classifications are known and are then used to predict the classification for samples for which the classifications are unknown. In the case of Patternizr, each example is a pair of crimes, and the classification is whether the two crimes are in a pattern together. The collection of examples used for training is known as a training set. Each model learns to perform classification by incorporating information from a set of attributes, known as features, which are selected to represent relevant characteristics about each pair of crimes. Much of the work of building a machine learning model is in identifying and representing features to enable the model to make accurate classifications. We call this step feature extraction. For Patternizr, as we discuss in the Sample Selection section, we chose samples for the training set in such a way that we improved information gain. Finally, we trained a specific type of machine learning model—a random forest—using standard techniques.

We used a separate model for each of three different crime types (burglaries, robberies, and grand larcenies) because these crime types have a sufficient corpus of prior manually identified patterns for use as training examples. This corpus consists of approximately 10,000 patterns between 2006 and 2015 from each crime type. In addition to manually identified official patterns, a portion of this corpus is built from complaint records where the same individual was arrested for multiple crimes of the same type within a span of two days. When these arrest groupings are included with manually identified patterns, each crime type has approximately 30,000 complaints, which are included in a pattern—a small portion of the 200,000–400,000 complaints for each crime type recorded over the same ten-year period. In this section, we describe how we constructed the features used in each of the three machine learning models; then discuss how we trained the models that were put into production. Figure 1 depicts the process by which these models were produced. Although we constructed separate models for each of the three crime types, the models rely on a nearly identical set of features and are trained in the same way.

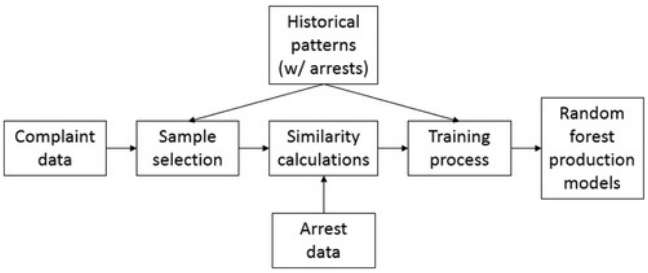

Fig. 1. Process overview for training Patternizr. Historical pattern data are used to choose a sample of complaints. That sample and any corresponding arrests are run through the similarity calculations to build features for learning. The random forest model takes those features and uses the historical patterns for training to produce a production model.

Feature Extraction

We construct the features on which each model relies from the information contained in complaints. A complaint contains a mix of unstructured text describing details of the crime and structured fields about the crime, including the date and time (which can be a range if the precise time of occurrence is unknown), location, crime subcategory, M.O., and suspect information. We use this information to calculate the five types of crime-to-crime similarities used as features by Patternizr: location, date-time, categorical, suspect, and unstructured text.

We selected these similarities by discussing pattern discovery with NYPD investigatory and enforcement personnel with many years of experience manually building patterns. They described the data sources they have used to identify patterns, as well as specific pieces of information that led them to believe that two complaints were connected. Additionally, they discussed many specific patterns they had identified and the unique set of circumstances that connected each group of crimes. In the course of these discussions, we also learned much about the inefficiencies that are inherent in manual pattern creation, as we detail in the introduction, and that could be reduced with the help of an automated tool.

Because Patternizr predicts the likelihood of a complaint pair belonging together in a pattern, the set of features read by the model (Table 1), which we call K, is derived from pairwise comparisons between crimes i and j. Each of these pairwise comparisons, which we call X_k, is a similarity calculation between a selected attribute k of the pair's complaint reports. The mathematical details of our more complicated features can be found in Appendix A.

For date-time, location, and categorical attributes, we used similarity computation methods previously described in related work on Cambridge Police Department data (15). However, we modified that formulation by excluding categorical information corresponding to administrative geographies (i.e., a feature noting that the pair exists in the same precinct) to encourage matches across such boundaries. Date-time and location attributes are useful features because detectives' experiences have shown that complaints in patterns are likely to happen nearby in space and time. Categorical attributes are important at identifying a consistent M.O. We also included metrics to compare the descriptions of groups of suspects

No.	Attribute	Type	Similarity measure
1	Location (XY)	Location	Distance apart (Euclidean)
2	Location (XY)	Location	Distance apart (exponential)
3	Location (XY)	Location	Binned midpoint (longitudinal)
4	Location (XY)	Location	Binned midpoint (latitudinal)
5	Location (XY)	Location	Location frequency (sum)
6	Location (XY)	Location	Location frequency (maximum)
7	Location (XY)	Location	Location frequency (minimum)
8	Location (XY)	Location	Location frequency (product)
9	Date-time of occurrence	Date-time	Time of day similarity
10	Date-time of occurrence	Date-time	Time of week similarity
11	Date-time of occurrence	Date-time	Size of occurrence windows (larger)
12	Date-time of occurrence	Date-time	Size of occurrence windows (smaller)
13	Date-time of occurrence	Date-time	Size of occurrence windows (difference)
14	Date-time of occurrence	Date-time	Days apart
15	Date-time of arrest	Date-time	Days between arrest and crime
16	Premise type	Categorical	Categorical similarity
17	Crime classification	Categorical	Categorical similarity
18	M.O. categorical fields	Categorical	Categorical similarity
19	M.O. weapon	Categorical	Categorical similarity
20	Location details	Categorical	Categorical similarity
21	Firearm discharged	Categorical	Categorical similarity
22	Crime attempted or completed	Categorical	Categorical similarity
23	Domestic crime indicator	Categorical	Categorical similarity
24	Medical assistance required	Categorical	Categorical similarity
25	Suspect height(s)	Suspect	Group similarity (continuous)
26	Suspect weight(s)	Suspect	Group similarity (continuous)
27	Suspect force used	Suspect	Group similarity (categorical)
28	Suspect count	Suspect	Difference
29	M.O. victim count	Suspect	Difference
30	Complaint narrative (burg only)	Unstructured text	Average similarity
31	Complaint narrative (burg only)	Unstructured text	Sum of similarity
32	Complaint narrative (rob and GL)	Unstructured text	Cosine similarity
33	M.O. suspect statement (rob and GL)	Unstructured text	Cosine similarity
34	M.O. victim actions (rob and GL)	Unstructured text	Cosine similarity
35	M.O. method of flight (rob and GL)	Unstructured text	Cosine similarity
36	Premise name (rob and GL)	Unstructured text	Cosine similarity
37	Property taken (rob and GL)	Unstructured text	Cosine similarity
38	All unstructured text	Unstructured text	Cosine similarity
39	Complaint narrative	Unstructured text	Rare-word matches

Table 1. List of the features included in Patternizr. These features are indexed by the variable k in the text. We use the following abbreviations: "burg" represents burglary, "rob" represents robbery, and "GL" represents grand larceny.

(e.g., the count of suspects who have committed a crime); however, we excluded sensitive suspect attributes (e.g., race). Suspect descriptions are particularly useful in identifying patterns of violent crimes such as robberies because, in such crimes, the victim physically encounters the suspect. Finally, we added features for the similarity of unstructured text narratives, including the cosine similarity of vector representations of the descriptions, as well as the number of rare-word matches between two descriptions. These features were included as a reflection of the central role that unstructured text plays in the manual identification of patterns.

Many features represent similar, but distinct, representations of the underlying data. This allows the algorithm to leverage as much available information as possible. We constructed three slightly different feature sets for each crime type, based on the characteristics relevant, available, and found to be important for that crime type. For example, the M.O. categorical field for "method of entry" is relevant to burglaries but not for robberies or grand larcenies.

Location Features. We incorporate eight similarities (labeled with indices $k = 1$ through $k = 8$) derived from location information, including a strict Euclidean distance d (in feet) between the crime pair and an exponential factor e^d. We also include two location features corresponding to longitude and latitude but binned to very coarse squares (over nine miles on a side, for a total of roughly 11 boxes possible across the geography of New York City) to minimize the possibility that higher-resolution geographies could be used by the model as a proxy variable for sensitive attributes (such as race). For each pair of crimes, we note which bin contains the pair's midpoint and include this as a feature. We also use a two-dimensional location frequency distribution calculated with kernel density estimation (KDE) (22), which represents a heatmap of where crime is most (and least) common in the city. For each pair, we add a feature for the sum, maximum, minimum, and product of these KDE values, providing the model with a range of information on how common the locations were for both crimes.

Date-Time Features. Seven separate similarities ($k = 9$ through $k = 15$) depend on the date-time attributes of the two complaints. The time-of-day similarity measure, which compares the time of day of the two crimes' occurrences, is drawn directly from Wang et al. (15). Following their approach, we calculate the frequency distribution of occurrence times for each crime type using KDE. Then, for each pair, we weight the difference between occurrence times by the frequency of those occurrence times. Crimes that occur close in time to each other, and at rare times, will have a low measure, indicating greater similarity between crime times. Crimes that occur far apart in time, and during common times, will have a high measure, indicating lower similarity between crime times. Alternatively, crimes that are a certain distance apart in time but occur during rare times will be more similar than crimes that are the same distance apart in time but occur during common times. This calculation can also be applied to occurrence times specified over an occurrence window—typical for crimes such as burglaries where the victim often is not present to witness the crime and therefore knows only that the crime happened within some time span. Comparisons across occurrence windows are made by dividing the window evenly into ten points and averaging similarities across all ten points. We applied the same process to time-of-week similarities. In addition, we included features for the duration of the larger and smaller occurrence windows from the pair of crimes, as well as the difference in size of the two occurrence windows. We included a simple feature—days apart—that we calculated between the midpoints of the time ranges if the occurrence time was over a range. Finally, we included a feature for the time between an arrest associated with the earlier crime and the occurrence of the later crime so that the model could have information about whether a criminal was potentially incarcerated.

Categorical Features. Much of the information in a complaint is in categorical form, including premise type, crime classification, the M.O. itself, weapon type, and details about the crime's location. Other information is stored in categorical fields that have yes/no/unknown structures; examples include whether a firearm was discharged during the crime, whether the crime was attempted or completed, whether someone required medical assistance, and whether the crime was domestic in nature. For all our categorical features ($k = 16$ through $k = 24$), we incorporate a simplified version of Goodall's similarity measure (23), identical to that used by Wang et al. (15). This measure accounts for the frequency of categorical values, causing two records that match on a rare value to be rated as more similar than two records that match on a common value. For categorical values that may take on multiple values for a single crime (e.g., multiple weapon types may be associated with a single robbery), possible values are encoded as dummy variables (22) before the categorical similarity calculation. In addition, many categories are too fine grained to be useful, because Goodall's simplified similarity measure is only nonzero when categories match exactly on both complaints. For example, NYPD crime classifications for burglaries include three separate categories for residential burglary, even though a perpetrator with the same M.O. may plausibly commit crimes across all three types. To address this issue, we asked uniformed police experts to group the existing classification into broader categories using domain knowledge. We included both the ungrouped and grouped crime classification similarities as features for the algorithm.

Suspect Features. We chose to include only nonsensitive suspect attributes, including height, weight, force used, and count of suspects, as suspect features for Patternizr. Suspect-attribute comparison differs from categorical comparison because it is often a many-to-many comparison for a crime pair. It is generally not possible to know from two lists of suspects which descriptions correspond to the same individual; therefore, comparisons must be made between all possible pairs of individuals (with each list providing one of the pair). For example, for the height and weight continuous variables ($k = 25$ and $k = 26$), we calculated the minimum difference between each possible pair of suspect descriptions and then divided the sum of differences by the quantity of suspects. For the categorical feature of force used (i.e., whether the suspect threatened to use force, or actually used force, in the commission of a robbery), we counted and tabulated the number of matches between possible suspect pairs ($k = 27$) in a similar fashion. We also included the difference in suspect count and victim count (which is a characteristic of the M.O.) as features ($k = 28$ and $k = 29$).

Unstructured Text Features. We also created a number of unstructured text features, including comparisons of the complaint text narrative, which plays a central role in the manual identification of patterns. Unstructured text also comprises the suspect's statements, victim actions, method of flight, premise name, and property taken. For burglaries, we calculate a word-by-word score identical to the categorical similarities, but on a word-by-word basis rather than attribute basis, for words that match between the narratives on both

complaints. We then provide an average and sum of these similarities ($k = 30$ and $k = 31$).

For robberies and grand larcenies, we use a more advanced method on other unstructured fields in addition to the narratives. We calculate the term-frequency/inverse-document frequency (TF/IDF) (24) vector, representing the presence and importance of the set of words that comprise each unstructured text. We compute a cosine similarity between the pair of TF/IDF vectors for each unstructured text attribute (labeled with indices $k = 32$ through $k = 37$).

We also combine all unstructured text from each complaint and treat it as a separate unstructured text attribute; this permits information that appears fragmented across different narratives to be compared at once across all possible sources. For this combined narrative, we calculate the cosine similarity ($k = 38$) and also count the number of rare words that did not appear in the training corpus but do appear in both complaints, and we include that count as a feature ($k = 39$). These unusual words may indicate the presence of a consistent and unusual M.O.

Sample Selection

A very small proportion of all possible crime pairs (about 8×10^{-8}, 3×10^{-8}, and 9×10^{-9} of possible burglary, robbery, and grand larceny pairs, respectively) are together in manually identified patterns. We refer to these crime pairs as positive pairs. To preserve maximum information gain from the corpus of patterns while also permitting model training within memory constraints, we included all positive pairs in the training set and performed down-sampling on the set of possible negative pairs.

Purely random sampling of negative pairs would largely consist of pairs that are far apart in space and time, encouraging the model to identify positives based solely on proximity in those dimensions. To counteract this effect, we sampled nearby negative pairs at a higher rate than distant negative pairs. We paired every pattern crime with a random sample of 2% of all negative examples within 3,500 feet and an additional random sample of 0.1% of all crimes within 80 days of the crime's midpoint in time. These were sampled uniformly: all negative examples within 3,500 feet or within 80 days of the crime's midpoint in time were equally likely to be selected. We repeatedly tested different thresholds until models trained on these data rated nonspatial or nontemporal features roughly as important as distance and time apart (as measured by feature importances). Finally, a substantial portion of distant negative pairs that included one crime known to be part of a pattern was also selected at random—approximately 20 times the volume of already selected positive and nearby pairs for that crime.

We subsequently uniformly sampled this set to fit in memory for model training. We trained the model for each crime type on 20–32 million pairs of crimes. For burglaries, robberies, and grand larcenies, 0.2%, 0.08%, and 0.06% of this reduced set of pairs were positive, respectively.

Model Training

For each of the three crime types, we relied on a random forest model to calculate an overall similarity score between two crimes. We chose a random forest model for several reasons. First, tree-based classification algorithms are able to learn interactions between features, which is important when considering how separate crime subtypes could exhibit different similarity structures. For example, grand larceny pickpocketing patterns may generally occur much closer in space (e.g., on a specific street corner) when compared with grand larceny shoplifting patterns, which may generally happen across the city. Second, random forests have demonstrated an ability to achieve impressive performance while avoiding overfitting (25). Third, random forests are implemented in many standard machine learning libraries, can be trained and executed in parallel, and do not require graphics processing units to execute. Finally, similar to many other classification algorithms, random forests generate a "score"; in Patternizr's case, this represents the confidence that the pair belongs together in a pattern. These similarity scores are scaled between 0 and 1; therefore, they can be naturally represented to an investigator in the form of a percentage.

We used the standard implementation of the random forest algorithm in the Python library scikit-learn (26). Each random forest model is an ensemble of hundreds of decision trees that are trained on the calculated features and responses. We trained each individual tree on an independently bootstrapped sample of the training set, with a randomized portion of the full set of features K. We describe the mathematical details of how we trained the trees in Appendix B.

Each random forest model also has a collection of settings, which are known as hyperparameters, that specify model structure and operation. We optimized several hyperparameters for the random forest model, including the size of the randomized subset of features and the maximum allowed tree depth. We used a cross-validated randomized hyperparameter search (27), which repeatedly selects a random combination of hyperparameters from a prespecified distribution of values and then tests the performance of this combination on several subsets of the data, preventing overfitting to any par-

Algorithm	Completely rebuilt patterns (no.)	Completely rebuilt patterns (%)	Patterns with match in top 10 (no.)	Patterns with match in top 10 (%)
Burglary	55 (43–67)	37% (29%-45%)	123 (114–132)	82% (76%-88%)
Robbery	39 (29–49)	26% (19%-33%)	117 (107–126)	78% (71%-84%)
Grand larceny	41 (30–51)	27% (20%-34%)	114 (103–124)	76% (69%-83%)

Table 2. Accuracy of Patternizr on 150 test patterns. The average performance is listed first, followed by the 95% bootstrapped confidence interval of the performance in parentheses.

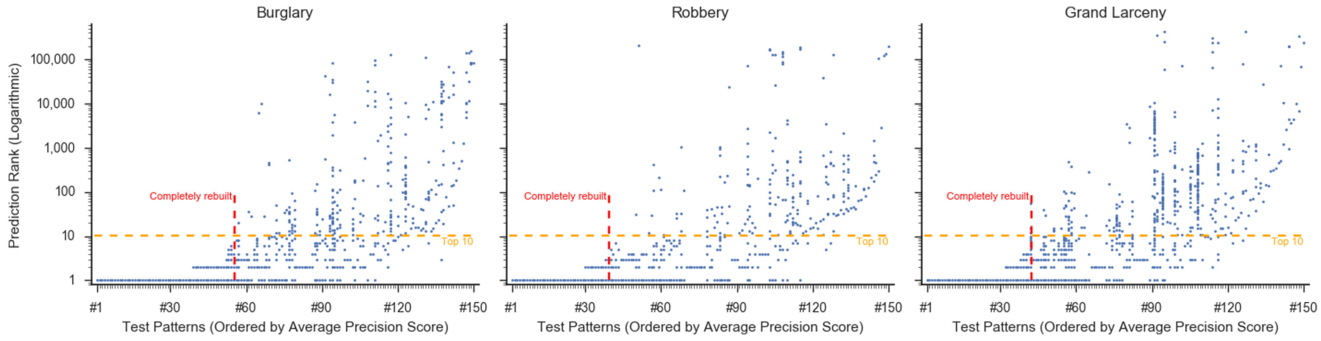

Fig. 2. Performance of Patternizr on 150 test patterns. Each position on the x axis corresponds to a test pattern, and each dot corresponds to the ranking of a crime in that pattern. Patterns are ordered left to right by the average precision score for that pattern as a visual aid. Patternizr did best on the patterns that appear to the left side of each chart. Dashed lines indicate the proportion of test patterns with complete reconstruction (left of the vertical red line) and the patterns with a match in the top ten rankings (under the horizontal orange line). Roughly one-third of the test patterns are completely reconstructed (as we also detail in Table 2); that is, no complaints outside of the pattern are ranked higher than a complaint included in the pattern, and roughly four-fifths of the test patterns have at least one of the complaints in the pattern ranked in the top ten.

ticular subset. For each crime type, we trained one random forest model with up to 220 trees using the generated features and tuned hyperparameters.

We set our hyperparameter search to optimize hyperparameters for the best possible average precision (28)—an evaluation metric that gauges whether desired matches are ranked highly in a query. In Patternizr's case, the query is the list of complaints returned by Patternizr when the user chooses a seed. Average precision relies on the precision metric, which measures the proportion of a predicted positive class that is actually positive. For Patternizr, precision is the percentage of suggested matches that are actually in a pattern with the seed. Each true match in Patternizr's ranked list can be represented by the precision at that rank—the number of true matches encountered so far divided by the rank (i.e., the total number of suggested matches provided by Patternizr at that point). Average precision, therefore, is simply an average of these precisions for all true matches in the test pattern.

Performance

We tested the performance of Patternizr on a variety of objectives. First, we measured the proportion of test patterns that were entirely or partially rebuilt by the models, even though the models had not previously seen these test patterns. Second, we constructed a benchmark model as a simplified proxy for existing practice to test whether investigators using Patternizr would be more efficient than those using existing methods. Third, we examined the features each model had isolated as most useful in identifying patterns and compared those with investigators' intuition. Finally, we measured whether Patternizr made consistent recommendations across racial groups.

We validated all three Patternizr models on official patterns that were not included in the training set. For each crime type, we randomly chose 150 test patterns and randomly selected a seed from each pattern. We then noted the rankings of the remaining pattern crimes as provided by Patternizr's similarity scores. We bootstrapped this process by randomly selecting these 150 test patterns 1,000 times, which provided reliable estimates and uncertainty bounds. Table 2 shows the results from this analysis.

The algorithm completely rebuilt (i.e., ranked all in-pattern pairs ahead of all nonrelated pairs) approximately one-third of test patterns. Additionally, approximately four-fifths of burglary, robbery, and grand larceny patterns had at least one match in the top ten rankings out of the entire corpus of complaints for that crime type. Figure 2 shows the rankings for all

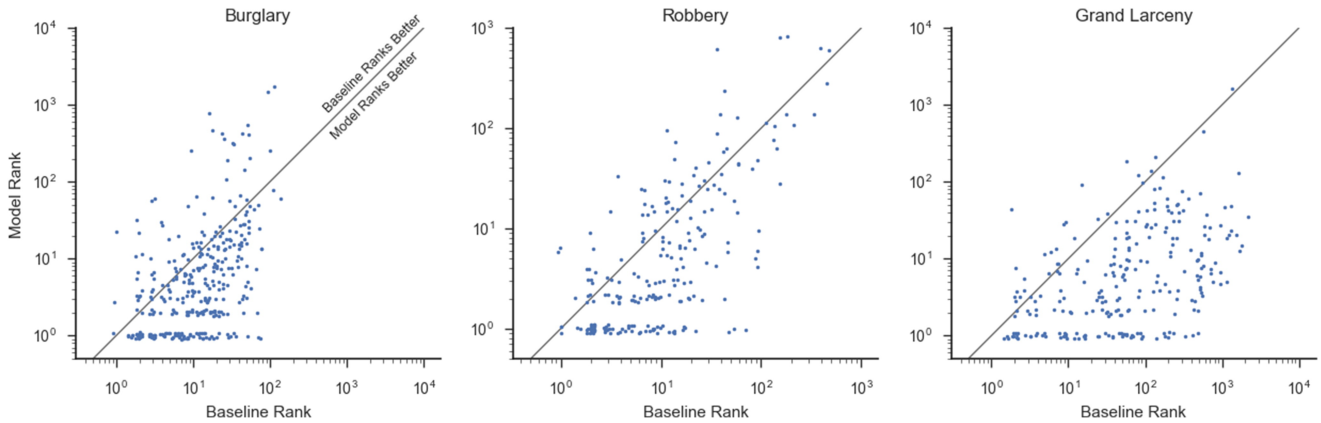

Fig. 3. Peformance of Patternizr against a simple baseline. Each dot represents a pair of complaints, where one half of the pair is a seed from the test set and the other half of the pair is another complaint in that pattern. Model performance is better than baseline performance for each pair of sample complaints that lies below the diagonal line; that is, the algorithm gives a ranking closer to the top of the list than the baseline algorithm.

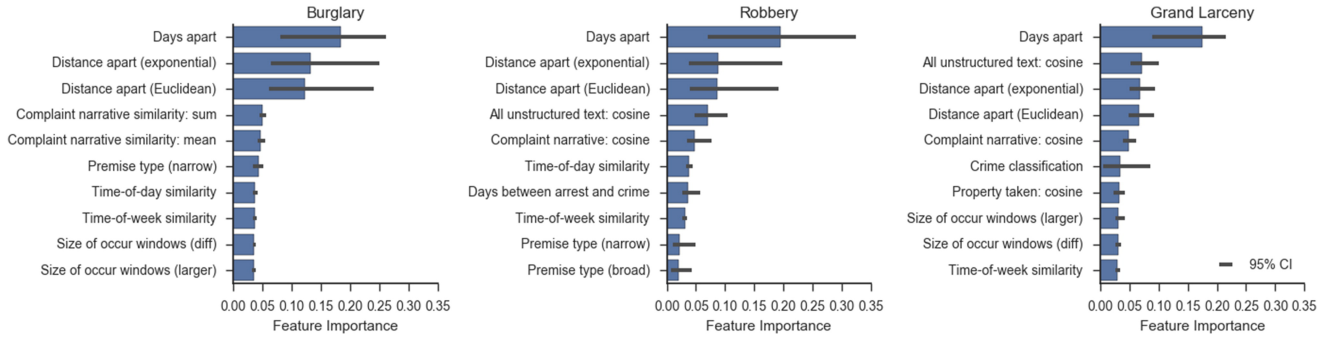

Fig. 4. Feature importances for Patternizr. Feature importance is one way to gauge the relative contribution of different features to the overall classification of a random forest. Feature importances are calculated by measuring the Gini impurity reduction at each node at which that feature is used, weighing these reductions by the number of samples that were routed to that node, and then averaging these calculations for each feature across all trees in the forest in which that feature is used.

test patterns. In reality, this performance is likely an underestimation, given that investigators have likely missed actual matches in the historical corpus.

To test whether Patternizr improves investigator efficiency, we built a simple baseline algorithm for comparison. The baseline algorithm assumes that investigators examine complaints nearest in time to the seed complaint first while always limiting themselves to complaints in the same precinct as the chosen seed. This method provides a baseline ranked list that we can compare with Patternizr's ranked results. We restricted the comparison with local patterns in the test set because the baseline algorithm would perform poorly (and unrealistically) on patterns that span multiple precincts. We compare the ranking of that complaint with how it appears in the ranked list generated by Patternizr for all the local patterns in our test data set (Figure 3).

We found that for all three crime types, Patternizr is typically an improvement on the baseline algorithm; this estimate is conservative because Patternizr is especially helpful on patterns spanning multiple precincts.

In Figure 4, we show the top ten feature importances for each model. The features for distance, days apart, and narrative information play a top role in all three models. NYPD's pattern subject matter experts confirmed that these three features contain key information for pattern creation. Other top features include similarity between other unstructured text fields and time-of-day and day-of-week similarity. None of the top ten features for each crime type was particularly surprising to our subject matter experts. Note that these calculations only measure importance; they do not indicate how positive or negative changes in the feature will affect the model score.

It is important to measure whether the recommendations that Patternizr makes are not only valuable but also fair, particularly given the growing concern that predictive policing tools may perpetuate disparate impact (29–31). We intentionally designed the algorithm to minimize disparate impact on any specific group. First, the algorithm is completely blind to sensitive information about potential suspects, including race and gender, which were not included as a similarity feature for the predictive model. Second, we kept potential proxy variables for sensitive information—particularly location—extremely coarse to ensure correlation with sensitive attributes had a very low degree of certainty while retaining some very general information about location. Finally, and most important, several levels of expert human review are still required to establish a pattern, minimizing the potential for a seemingly likely (but incorrect) recommendation to result in any enforcement action.

To confirm that these precautions were effective, we tested whether any of the three models recommended pairs with a certain racial group at a different frequency than either random pairing or existing identified patterns. We randomly selected 5–10 million test pairs from each crime type and drew 1,000 bootstrapped samples from each of these samples to estimate uncertainty (Figure 5).

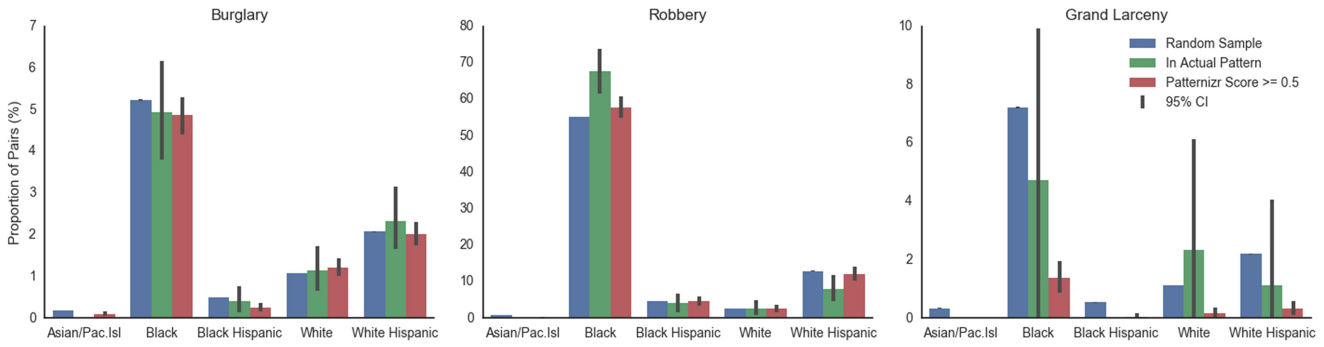

Fig. 5. Proportion of pairs by race for random, historical pattern, and Patternizr-recommended pairs. Confidence intervals are calculated from the range of values that appear in 1,000 bootstrapped samples. Racial categories correspond to the official descriptions listed on NYPD complaint forms. Notably, many types of grand larceny will not contain suspect information because there were no witnesses to the crime, thus producing relatively higher levels of uncertainty for grand larceny than for the other two crime types.

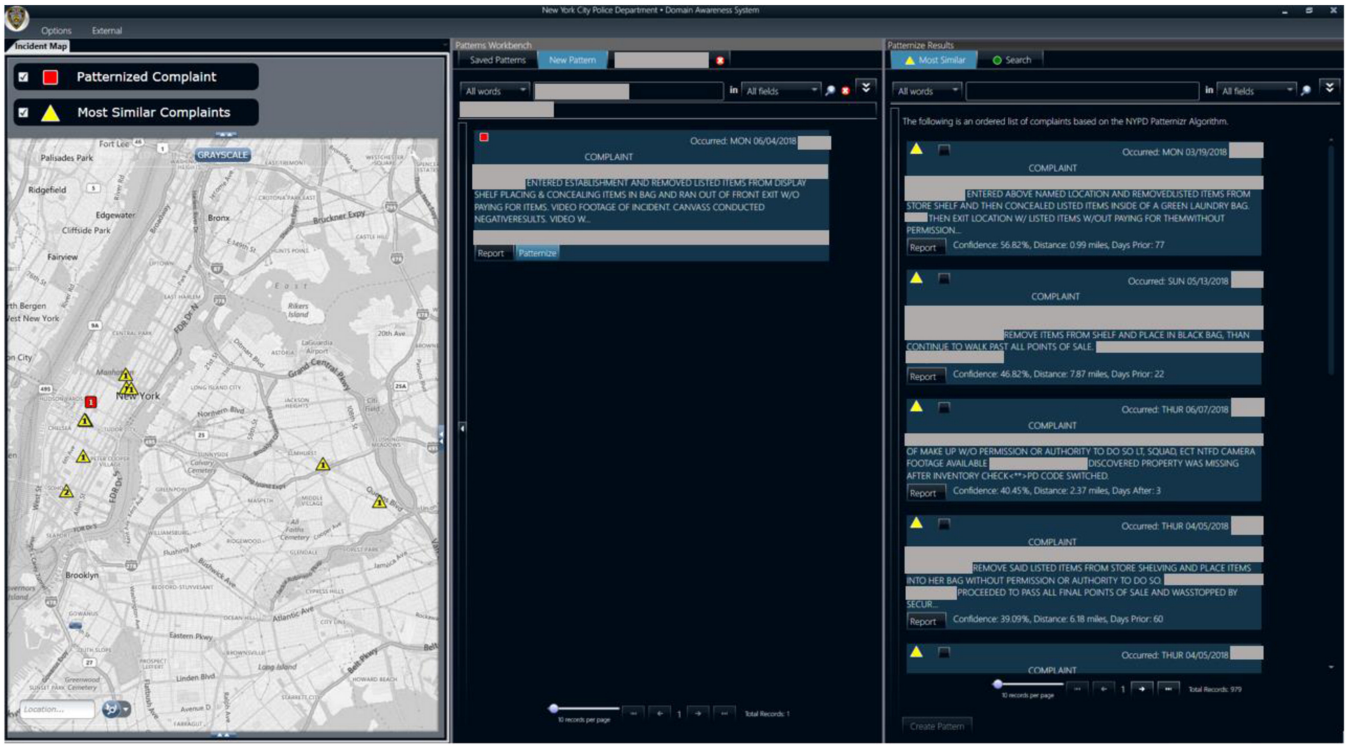

Fig. 6. Screenshot of Patternizr in the NYPD Domain Awareness System. The map of the seed and most similar complaints is in the left panel, the seed complaint is in the middle panel, and the most similar complaints are in the right panel. Grey boxes obscure law enforcement sensitive and personally identifiable information.

For all three crime types, the proportion of Patternizr pairs with a similarity score greater than 0.5, which included a suspect description of any race, was either consistent within the 95% confidence limit or lower than a random sampling of all pairs. That is, this analysis uncovered no evidence that Patternizr recommends any suspect race at a higher rate than exists with random pairing.

Implementation

We incorporated finalized models into the back end of the department's Domain Awareness System (DAS), a citywide network of sensors, databases, devices, software, and infrastructure (9, 21). We processed all historical pairs of complaints in parallel in the cloud, against ten years of records for burglaries and robberies, and against three years of grand larcenies. This historical load took 19.4 days on approximately 1,600 cores. For new and revised complaints, similarity scores are calculated and updated three times each day. Each new or revised crime is scored against the entire corpus of crimes before it is incorporated into DAS for querying by users throughout the department.

Officers and analysts can access Patternizr through the custom software application, which is a part of DAS on NYPD desktop computers. To use Patternizr, an investigator presses a "Patternize" button on any seed complaint to retrieve all available calculated similarity scores. Investigators are presented with a rank-ordered list of results, from most to least similar to the seed, with ten results displayed at a time. For each result, the distance, time apart, and algorithm-calculated similarity score are all displayed. A map displays the seed and similar results (Figure 6), and users can read a few important details about each complaint before choosing to examine the seed and result complaints side by side in full detail. The use of a ranked list, as opposed to a similarity score threshold, allows us to always display the most similar crimes; thus, the analysts always have a starting point for their analysis.

The user interface gives investigators the ability to collaborate with the algorithm using their specialized expertise. If an investigator has reason to believe that a certain M.O. is particularly relevant for a given pattern (e.g., only burglarizing tools from construction sites), the investigator can actively filter and query the result list. A general text search, and filters for distance, time apart, presence of arrest, and premise type, are all available and may be used simultaneously. The remaining results are still sorted by Patternizr's similarity score. The filters do not reorder the results; they only exclude crimes that do not meet the search criteria. In this way, investigators can both search for a specific M.O. and utilize the rank order of the results list to immediately display the most similar results, given the manually specified M.O. criteria.

After approximately two years of development, including work on the algorithm, backend systems, and the user interface, Patternizr was deployed to production in December 2016. In parallel, throughout 2017, the NYPD hired a team of approximately 100 civilian analysts, deployed in the field, to perform crime analysis. These new crime analysts were trained to use Patternizr as part of their daily routine. For example, a crime analyst assigned to a precinct typically reviews crimes that occurred in his or her precinct to see whether they are part of a pattern. The uniformed offi-

cer that had been assigned this task before the crime analysts were hired used manual methods to perform this review; this review is now conducted using Patternizr.

Between January 2018 (with crime analysts at fully staffed levels) to the time of this paper's writing (July 2018), approximately 400 complaints per week were run through Patternizr, and this number is trending upward. This represents approximately 30% of all the burglaries, robberies, and grand larcenies recorded by the NYPD during that period. Approximately 80% of the usage comes from the new crime analysts, with the remaining 20% coming from uniform officers performing crime analysis and investigatory duties. Additionally, crime analysts created more than 90 preliminary patterns per week in the same period using Patternizr in DAS. These patterns are considered preliminary because they have not yet been reviewed and approved by the relevant specialist units; however, the volume of preliminary patterns created with Patternizr demonstrates the value of the tool. Debra Piehl, the NYPD's Senior Crime Analyst, says of the tool, "Patternizr dramatically improves efficiency compared to traditional methods, and it still allows the analysts that work for me to apply their own thinking and analysis. The science doesn't overwhelm the art" (32).

Crime analysts have also provided specific instances in which official patterns were created because the Patternizr algorithm recognized similarities between crimes that the analysts would otherwise have found difficult to notice. For example, one analyst was examining a recent robbery in her precinct; the perpetrator was shoplifting power drills from a commercial establishment and, when confronted, attacked and injured an employee with a hypodermic needle. The analyst ran the complaint through Patternizr, and the algorithm returned an additional robbery in a distant precinct where the perpetrator was shoplifting a drill and threatened an employee with a hypodermic needle. This robbery was likely identified by the algorithm because of high similarity scores for the time of occurrence, the robbery subtype (began as shoplifting), the weapon used, and several matching words in the narratives (e.g., drill, needle). The investigators combined these two complaints into an official pattern, along with two other larcenies committed by the same perpetrator, and the pattern was then passed to the detective squad. The NYPD conducted an investigation and arrested the perpetrator, who later pled guilty to larceny and felony assault and is currently awaiting sentencing.

A second example involves the discovery of a grand larceny of unattended-property pattern. In this example, an analyst was examining the theft of a watch from a locker at a hotel gym in Midtown Manhattan in the late afternoon. The analyst used Patternizr on this complaint and discovered three other complaints with the same M.O.—jewelry and watches were removed from a locker in a fitness facility in the late afternoon or early evening in the same area. The algorithm likely recognized similarities in the complaints' times of occurrence, their geographic proximity, the property removed (jewelry and watches), the premise types (gyms), and key words in the narrative (e.g., locker). Two suspects were identified through the review of video footage, and the NYPD's investigation is ongoing.

The helpful feedback we have received from our users has highlighted potential improvements to Patternizr. For example, users have requested that the grand larceny algorithm also calculate similarities to petit larceny complaints, which differ from grand larcenies only in the value of the items stolen. Users have also requested the ability to compare across crime types, such as robberies and grand larcenies, where the only difference is whether force was used. We are considering adding this functionality when we deploy a second version of the Patternizr algorithms.

Conclusion

Patternizr is a new, effective, and fair recommendation engine deployed by the NYPD to help investigators group related crimes. To our knowledge, this is the first time such an algorithm has been deployed to production in a law enforcement environment. It joins a growing list of machine learning applications customized for public safety and for the public sector in general. Patternizr is also one of a long line of data-driven tools prototyped and developed at the NYPD, including CompStat (7). These tools, when used properly, encourage precision policing approaches instead of widespread, heavy-handed enforcement techniques and enable investigators to focus on the art of policing instead of rote and uninteresting work. We expect that other departments, and local agencies in general, will continue this line of research by extracting similar value from existing government data.

Bibliography

1. Walt L Perry, Brian McInnis, Carter C Price, Susan C Smith, and John S Hollywood. *Predictive policing: The role of crime forecasting in law enforcement operations*. Rand Corporation, 2013.
2. Andrew Guthrie Ferguson. *The rise of big data policing: Surveillance, race, and the future of law enforcement*. NYU Press, 2017.
3. Greg Ridgeway. Policing in the era of big data. *Annual Review of Criminology*, 1:401–419, 2018.
4. Richard C Larson. *Urban police patrol analysis*, volume 28. MIT Press Cambridge, MA, 1972.
5. Linda V Green and Peter J Kolesar. Improving emergency responsiveness with management science. *Management Science*, 50(8):1001–1014, 2004.
6. Jack Maple. *The crime fighter: How you can make your community crime free*. Broadway, 2000.
7. Vincent E Henry. *The COMPSTAT paradigm: Management accountability in policing, business, and the public sector*. Looseleaf Law Publications, 2002.
8. George O Mohler, Martin B Short, Sean Malinowski, Mark Johnson, George E Tita, Andrea L Bertozzi, and P Jeffrey Brantingham. Randomized controlled field trials of predictive policing. *Journal of the American statistical association*, 110(512):1399–1411, 2015.
9. ES Levine, Jessica Tisch, Anthony Tasso, and Michael Joy. The New York City Police Department's Domain Awareness System. *Interfaces*, 47(1):70–84, 2017.
10. Richard Berk and Justin Bleich. Forecasts of violence to inform sentencing decisions. *Journal of Quantitative Criminology*, 30(1):79–96, 2014.
11. Jessica Saunders, Priscillia Hunt, and John S Hollywood. Predictions put into practice: a quasi-experimental evaluation of Chicago's predictive policing pilot. *Journal of Experimental Criminology*, 12(3):347–371, 2016.
12. Nelson Lim, Carl Matthies, Brian Gifford, and Greg Ridgeway. *To protect and to serve: enhancing the efficiency of LAPD recruiting*, volume 881. Rand Corporation, 2009.
13. David Weisburd, Elizabeth R Groff, Greg Jones, Breanne Cave, Karen L Amendola, Sue-Ming Yang, and Rupert F Emison. The Dallas patrol management experiment: can AVL technologies be used to harness unallocated patrol time for crime prevention? *Journal of Experimental Criminology*, 11(3):367–391, 2015.
14. Jennifer Helsby, Samuel Carton, Kenneth Joseph, Ayesha Mahmud, Youngsoo Park, Andrea Navarrete, Klaus Ackermann, Joe Walsh, Lauren Haynes, Crystal Cody, et al. Early intervention systems: Predicting adverse interactions between police and the public. *Criminal justice policy review*, 29(2):190–209, 2018.
15. Tong Wang, Cynthia Rudin, Daniel Wagner, and Rich Sevieri. Learning to detect patterns of crime. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 515–530. Springer, 2013.

16. Michael D Porter. A statistical approach to crime linkage. *The American Statistician*, 70(2): 152–165, 2016.
17. Robert M Bell and Yehuda Koren. Lessons from the Netflix prize challenge. *SiGKDD Explorations*, 9(2):75–79, 2007.
18. New York City Police Department. Citywide seven major felony offenses, 2000–2016, 2017. Accessed: 2017-12-20. http://www1.nyc.gov/assets/nypd/downloads/pdf/analysis_and_planning/seven-major-felony-offenses-2000-2016.pdf.
19. Marvin E Wolfgang, Robert M Figlio, and Thorsten Sellin. *Delinquency in a birth cohort*. University of Chicago Press, 1987.
20. Samantha L Gwinn, Christopher W Bruce, Steven R Hick, and Julie P Cooper. *Exploring crime analysis: Readings on essential skills*. International Association of Crime Analysts, 2008.
21. ES Levine and JS Tisch. Analytics in action at the New York City Police Department's Counterterrorism Bureau. *Military Operations Research*, 19(4):5–14, 2014.
22. Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: prediction, inference and data mining*. Springer-Verlag, New York, 2009.
23. Shyam Boriah, Varun Chandola, and Vipin Kumar. Similarity measures for categorical data: A comparative evaluation. In *Proceedings of the 2008 SIAM international conference on data mining*, pages 243–254. SIAM, 2008.
24. James H Martin and Daniel Jurafsky. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Pearson/Prentice Hall Upper Saddle River, 2009.
25. Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
26. Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in Python. *Journal of machine learning research*, 12(Oct): 2825–2830, 2011.
27. James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.
28. Christopher D Manning, Raghavan Prabhakar, and Schütze Hinrich. *Introduction to information retrieval*. Cambridge University Press, 2008.
29. Kristian Lum and William Isaac. To predict and serve? *Significance*, 13(5):14–19, 2016.
30. Sam Corbett-Davies, Emma Pierson, Avi Feller, Sharad Goel, and Aziz Huq. Algorithmic decision making and the cost of fairness. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 797–806. ACM, 2017.
31. Andrew Guthrie Ferguson. Policing predictive policing. *Wash. UL Rev.*, 94:1109, 2016.
32. Debra Piehl. Conversation with Evan Levine, June 22, 2018.
33. Daniel Jurafsky and James H Martin. Speech and language processing: An introduction to speech recognition, computational linguistics and natural language processing. *Upper Saddle River, NJ: Prentice Hall*, 2008.

Acknowledgements

The authors acknowledge the support of the men and women of the New York City Police Department. In particular, the authors thank Police Commissioner James P. O'Neill, the former Police Commissioner William J. Bratton, Chief Lori Pollock, Chief Dermot Shea, Deputy Chief Timothy McCormack, Inspector Anthony Tasso, Deputy Inspector Brandon del Pozo, Captain John Hall, Lt. Juan Duque, Lt. Phillip McEneaney, Sgt. Christopher Bray, Sgt. Joseph Klubnick, Sgt. Sean McCarthy, Deputy Commissioner Jessica Tisch, Assistant Commissioner Richard Schroeder, Assistant Commissioner Ronald Wilhelmy, Director Martha Norrick, Daniel Montiel, and Debra Piehl. Special thanks to Tong Wang, Cynthia Rudin, Lt. Daniel Wagner, Theo Damoulas, Ravi Shroff, and Joe McLaughlin.

Alex Chohlas-Wood's current affiliation is Stanford University, Stanford, CA.

Appendix A: Feature Extraction

In this section, we provide the mathematical details involved in our feature extraction process.

Categorical Attributes (*k = 16* through *k = 24*). Define N to be the number of documents in the corpus. First, we count the number of documents in the corpus where attribute k has the value x; we label these counts n_x^k. Then,

$$X_{i,j}^k = \begin{cases} 1 - \sum_{y \ s.t. \ n_y^k \leq n_{\text{match}}^k} \frac{n_y^k(n_y^k-1)}{N(N-1)}, & \text{if documents } i \text{ and } j \text{ match on attribute } k; \\ 0, & \text{otherwise.} \end{cases}$$

where the document count n_{match}^k is defined as the number of documents in the corpus that have the same value of attribute k as documents i and j (23).

Suspect Height and Weight Attributes (*k = 25* and *k = 26*). Call the minimal absolute difference between the value of a continuous variable associated with suspect h of complaint i to that of any suspect in complaint j $a_{i,j,h}$. In mathematical form, using the weight attribute $w_{i,h}$ of suspect h in complaint i as an example,

$$a_{i,j,h} = \min_f \Big[|w_{i,h} - w_{j,f}| \Big],$$

where the minimum function runs over all suspects f in complaint j. For example, if complaint 1 has suspects of weights 140 pounds (lbs), 180 lbs, and 220 lbs, and complaint 2 has suspects of weights 150 lbs and 215 lbs, then $a_{1,2,1} = 10$ lbs, $a_{1,2,2} = 30$ lbs, and $a_{1,2,3} = 5$ lbs. After calculating these comparisons, we transform them into a similarity via

$$X_{i,j}^k = \max\Big[\exp\Big(-P_i \sum_{h=0}^{P_i} \frac{a_{i,j,h}}{P_i + P_j}\Big), \exp\Big(-P_j \sum_{h=0}^{P_j} \frac{a_{j,i,h}}{P_i + P_j}\Big) \Big],$$

where P_i is the number of suspects for complaint i.

Suspect Force Used (*k = 27*). In this case, $a_{i,j,h}$ is just the number of suspects in complaint j with the same value of the categorical feature as suspect h in complaint i:

$$X_{i,j}^{k=27} = \max\Big[P_i \sum_{h=0}^{P_i} \frac{a_{i,j,h}}{P_i + P_j}, P_j \sum_{h=0}^{P_j} \frac{a_{j,i,h}}{P_i + P_j} \Big].$$

Unstructured Text for Robberies and Grand Larcenies (*k = 32* Through *k = 37*). Treating each attribute k separately, let the term frequency $tf_{x,i}$ be the count of word x in document i, let the document frequency df_x be the number of documents in the corpus containing word x, and let M be the number of unique words in the corpus. Then, for each word x, we calculate the inverse document frequency (33):

$$idf_x = \log\Big(\frac{N}{df_x}\Big) + 1.$$

The "+1" in this equation ensures that words that occur in every document in the training corpus will not receive a zero for inverse document frequency, which would preclude those words from contributing to the eventual cosine similarity. For each word x and document i, we calculate

$$s_{x,i} = tf_{x,i} \times idf_x.$$

We then build the TF/IDF vector S_i for each document i:

$$S_i = \Big[s_{0,i}, ..., s_{M,i}\Big].$$

Finally, for each pair of documents i and j, we calculate the cosine similarity between the two TF/IDF vectors:

$$X^k_{i,j} = \frac{S_i \times S_j}{S_i S_j}.$$

Appendix B: Random Forest Model Training

Trees are recursively built by identifying the best feature k and split s at each node m, which minimize the Gini index, a measure of node purity for the two descendent nodes (22). Let R be the subset of feature values X_k left or right of the split s, and let $p_{q,m}$ be the proportion of class q in node m. Then,

$$R_{\text{left}}(k,s) = \Big\{X|X^k \leq s\Big\}, R_{\text{right}}(k,s) = \Big\{X|X^k > s\Big\},$$

$$\hat{p}_{q,m} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = q),$$

where $\hat{p}_{q,m}$ is the estimated sample proportion of class q in left or right node m, N_m is the number of samples in node m, and $I(y_i = q)$ is an indicator function that returns 1 when sample i belongs to class q and returns 0 otherwise. We set q = 1 (i.e., positive examples) and find the combination of k and s that minimizes the Gini index as follows:

$$\min_{k,s} \Big[2\Big(\hat{p}_{1,\text{left}}\big(1-\hat{p}_{1,\text{left}}\big)\Big) + 2\Big(\hat{p}_{1,\text{right}}\big(1-\hat{p}_{1,\text{right}}\big)\Big)\Big].$$

In addition, the group of features K_m available for splitting at each node is chosen from a randomly selected subset of all available features K. Some or all of these features may be available for splitting at each node; from these available features, the model selects the feature and split that best separate the child nodes into purer groups of either in-pattern or unrelated crime pairs.