San Francisco Crime Classification

Gianmarco Donetti
Politecnico di Milano, DEIB
Milan, Italy
gianmarco.donetti@mail.polimi.it

ABSTRACT

As someone could tell, black clouds and silver linings. Exploiting a not so happy and glad scenario as the crime occurrences in San Francisco, the goal of this paper is to apply and analyze the results of some machine learning techniques in order to estimate the category of crimes occurred in the city of the Golden Gate Bridge. After a formal and mathematical formulation of the problem, a nice preliminary exploration is proposed, followed by the adopted methodologies description and brief discussions on the results obtained, showing how Naïve Bayes classifier for multivariate Bernoulli models and Logistic Regression can outperform in this probability estimation case.

KEYWORDS

machine learning, classification, probability estimation, supervised learning

1. INTRODUCTION

From 1934 to 1963, San Francisco was infamous for housing some of the world's most notorious criminals on the inescapable island of Alcatraz. Today, the city is known more for its tech scene than its criminal past. But, with rising wealth inequality, housing shortages, and a proliferation of expensive digital toys riding BART to work, there is no scarcity of crime in the city by the bay. From Sunset to SOMA, and Marina to Excelsior, this competition's dataset provides nearly 12 years of crime reports from across all of San Francisco's neighborhoods. Given time and location, we try to predict the category of crime that occurred, in a probabilistic way that tells us for each different crime the probability to belong to each class.

2. PROBLEM FORMULATION

This section begins with a briefing on the data representation, followed by a discussion of problem definition and a first preliminary exploration of the data.

2.1 Data Representation

The goal of this machine learning problem is to predict the category of crimes that occurred in the city by the bay. The given dataset contains incidents derived from SFPD Crime Incident Reporting system. The data ranges from 1/1/2003 to 5/13/2015. The training set and test set rotate every week, meaning week 1,3,5,7... belong to test set, week 2,4,6,8 belong to training set.

Then, we can call C the set of all the crimes reported in our dataset. We can further distinguish between C_{TRAIN} and C_{TEST}. Each specific incident event $c_i \in C$ is a tuple, with n attributes in the case of train samples, $n\text{-}3\text{+}1$ in the case of test samples. These attributes are:

1. *Dates* - timestamp of the crime incident
2. *Category* - category of the crime incident (only for the train data samples). This is the target variable you are going to predict
3. *Descript* - detailed description of the crime incident (only for the train data samples)
4. *DayOfWeek* - the day of the week
5. *PdDistrict* - name of the Police Department District
6. *Resolution* - how the crime incident was resolved (only for the train data samples)
7. *Address* - the approximate street address of the crime incident
8. *X* - Longitude
9. *Y* – Latitude

We can see that the minus three in the dimensionality of test samples is due to the lack of three attributes present in the training ones, while the plus one is due to the labelling of each tuple with an *Id* field. Then, every $c_i \in C_{TRAIN}$ is 9-dimensional, while every $c_i \in C_{TEST}$ is 7-dimensional. In the end, the sizes of the dataset are: $m_{TRAIN} = 878049$ and $m_{TEST} = 884262$.

2.2 Problem Definition

Machine Learning speaking, the ultimate aim of this study is to build a model h, which performs a probability estimation on each test sample of the *Category* attribute. Then, for each new crime and for each category class, it should propose the probability of being included in that specific class. Trivially, the summation of all the probability for each sample must summed up to *1.0*, otherwise a normalization step is applied.

Finally, writing all these reflections in a mathematical way, we call A the set of attributes names belonging to each test sample (i.e., $A = <$*'Id', 'Dates', 'DayOfWeek', 'PdDistrict', 'Address', 'X', 'Y'*$>$), and V the set of all the possible classes of crime category, we have to find the best model h that, for each $c_i \in C_{TEST}$, for each $v_z \in V$, estimates the probabilities:

$$P\left(c_i^{Category} = v_z \mid \{c_i^j\}, \forall j \in A\right)$$

The loss function to minimize is the multi-class version of the Logarithmic Loss. Each observation is in one class and for each observation we submit a predicted probability for each class. This is the function:

$$logloss = -\frac{1}{M}\sum_{i=1}^{M}\sum_{z=1}^{Z} y_{i,z}\log\left(p_{i,z}\right) \text{ [1]}$$

where M is the number of observations, Z is the number of class labels, $y_{i,z}$ is 1 if observation i is in class z and 0 otherwise, and $p_{i,z}$ is the predicted probability that observation i is in class z.

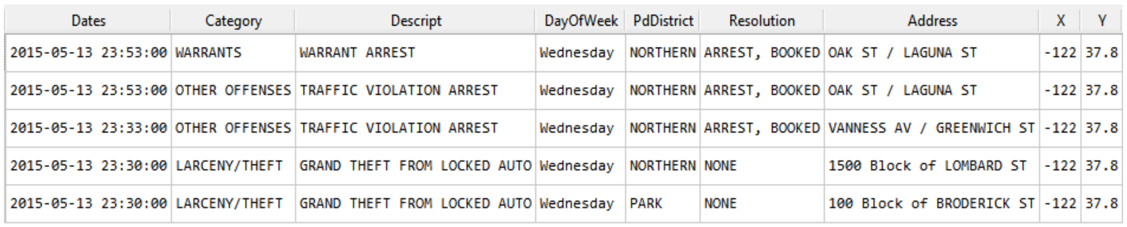
2.3 Preliminary Exploration

As every machine learning problem, before trying to predict or estimate anything, the best thing to do is to have a look at the data. In this step, I try to pull out some nice plots where we can have a better idea of the problem itself and we can try to see the first correlations between attributes.

Opening the first rows of the training data set, we can see in the table of Figure 1 all the attributes and their format. In particular, we see the datetime format of the *Dates*, three examples of *Category* classes, two examples of type of *Resolution*, and the geographical references of the *Address* fields, the *X*- and the *Y*-fields. Analyzing the occurrences of the different *Category* classes, we have a dominance of various larcenies, assaults, drug-related crimes and vandalisms acts, as Figure 2 reported.

Now, I make an analysis considering only the top occurred crimes category classes, that are '*Larceny/Theft', 'Assault', 'Drug/Narcotic', 'Vehicle theft', 'Vandalism'* and *'Burglary'* and perform some aggregated studies, grouping the crime events with different time granularities. The groupings by are per hour, per day of the week, per month, per year, and eventually per year & month, visualizing the complete historical trend of crimes in the data set. All the results are shown in Figure 3, from (a) to (e).

The next groupings in the data are considering two different features: the first one (Figure 4) takes into account the Police Department of reference, while the second one (Figure 5) takes into whether in the field *Address* the word ‘Block’ appears or not. Relevant result could be retrieved from all these groupings.



Dates	Category	Descript	DayOfWeek	PdDistrict	Resolution	Address	X	Y
2015-05-13 23:53:00	WARRANTS	WARRANT ARREST	Wednesday	NORTHERN	ARREST, BOOKED	OAK ST / LAGUNA ST	-122	37.8
2015-05-13 23:53:00	OTHER OFFENSES	TRAFFIC VIOLATION ARREST	Wednesday	NORTHERN	ARREST, BOOKED	OAK ST / LAGUNA ST	-122	37.8
2015-05-13 23:33:00	OTHER OFFENSES	TRAFFIC VIOLATION ARREST	Wednesday	NORTHERN	ARREST, BOOKED	VANNESS AV / GREENWICH ST	-122	37.8
2015-05-13 23:30:00	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Wednesday	NORTHERN	NONE	1500 Block of LOMBARD ST	-122	37.8
2015-05-13 23:30:00	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Wednesday	PARK	NONE	100 Block of BRODERICK ST	-122	37.8

Figure 1: First five rows from the training data set.

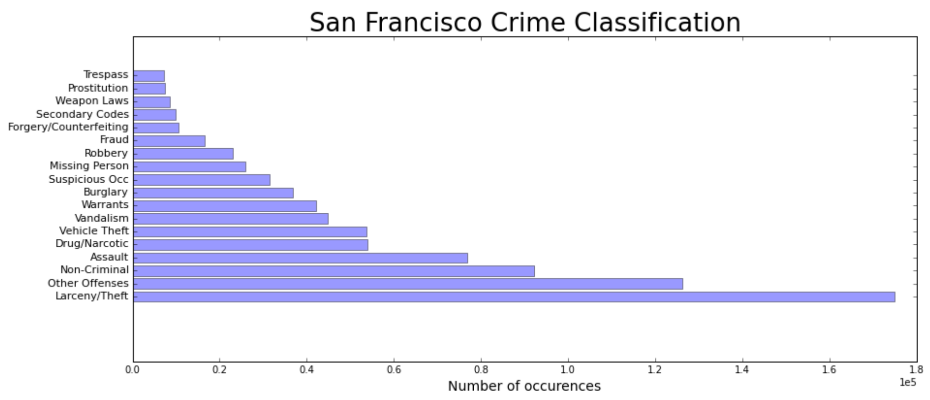
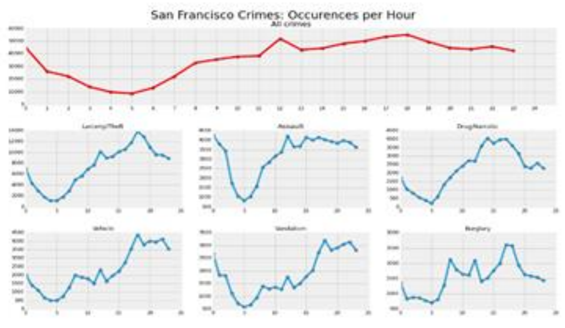
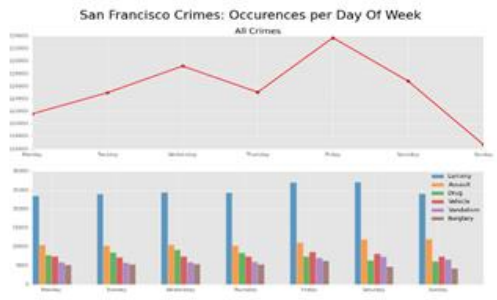


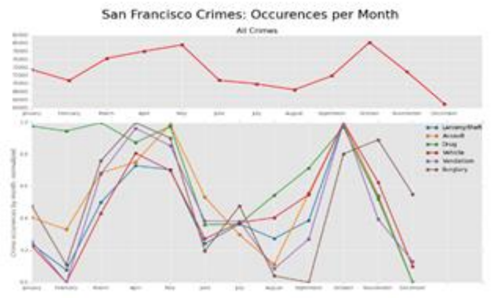
Figure 2: The 18 most occurred crimes category classes.



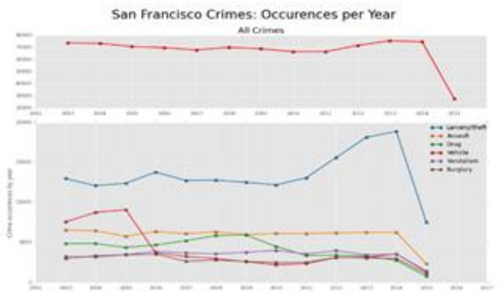
(a) Crime occurrences grouped by hour



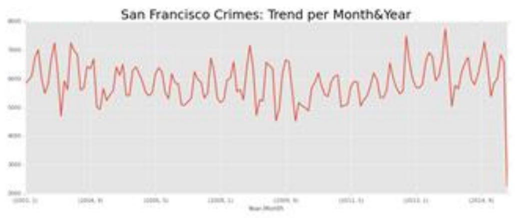
(b) Crime occurrences grouped by day of the week



(c) Crime occurrences grouped by month



(d) Crime occurrences grouped by year



(e) Crime occurrences grouped by year and month

Figure 3: Crime occurrences aggregated with different time granularities.

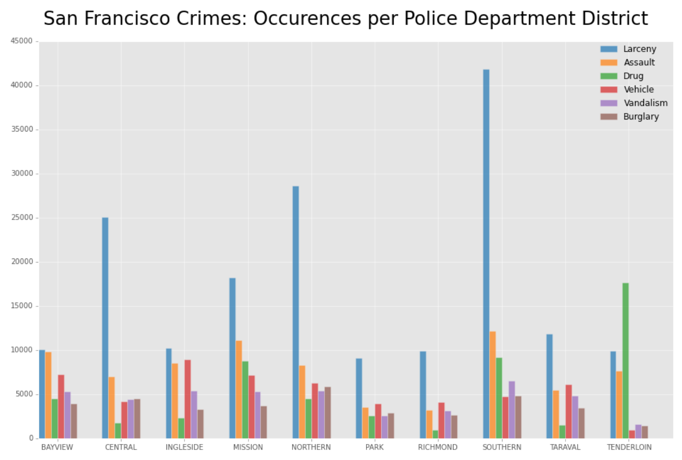


Figure 4: Crime occurrences aggregated per Police Department.

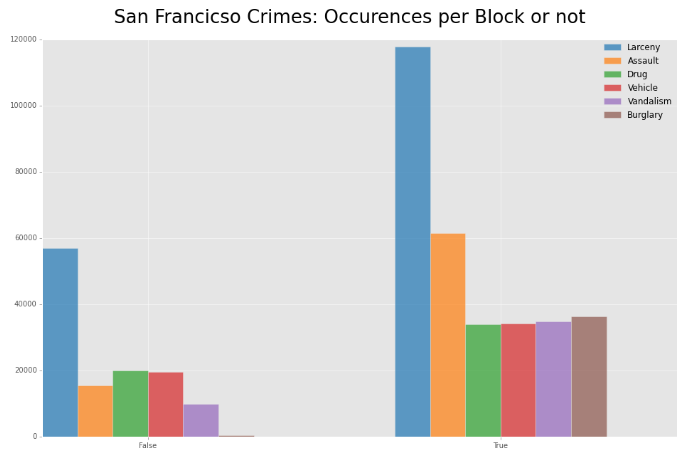


Figure 5: Crime occurrences aggregated per type of address.

3. METHODOLOGY

Different techniques have been adopted in order to find a good solution to the problem. In this section I propose brief summaries for all of them.

3.1 Naive Bayes classifier for multivariate Bernoulli models

The first method I approached is a Naïve Bayes classifier, specific for the case of multivariate Bernoulli models. It implements the naive Bayes training and classification algorithms for data that is distributed according to multivariate Bernoulli distributions; i.e., there may be multiple features but each one is assumed to be a binary-valued (Bernoulli, boolean) variable. Therefore, this class requires samples to be represented as binary-valued feature vectors; if handed any other kind of data, a Bernoulli Naïve Bayes instance may binarize its input (depending on the binarize parameter).

The decision rule for Bernoulli naive Bayes is based on:

$$P(x_i|y) = P(i|y)x_i + (1 - P(i|y))(1 - x_i) \text{ [2]}$$

3.2 Logistic Regression

The second model I approached comes from the linear ones, that is regularized Logistic Regression, which is very specific for probability estimation problem. In this model, the probabilities describing the possible outcomes of a single trial are modeled using a logistic function. The used implementation can fit a multiclass (with one-vs-rest paradigm) logistic regression with optional regularization, implied by the parameter *C*.

As an optimization problem, binary class norm-2 penalized logistic regression minimizes the following cost function:

$$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{m} \log(\exp\left(-y_i\left(X_i^T w + c\right)\right) + 1) \text{ [3]}$$

3.3 Bagging

In order to reduce the high variance, induce by complex model, in some case I performed a bagging phase [4]. In ensemble algorithms, bagging methods form a class of algorithms which build several instances of a black-box estimator on random subsets of the original training set and then aggregate their individual predictions to form a final prediction. The number of estimators used is always of 10.

3.4 Boosting

Also the boosting of simple methods has been performed. The chosen model follows the popular boosting algorithm AdaBoost, introduced by Freund and Schapire [5].

An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases. The used class implements the algorithm known as AdaBoost-SAMME [6].

3.5 Other approaches

In order to try other algorithms and methods, I stressed our data with two other models, but the results were not so satisfactory, from the point of view of scalable time performance. Indeed, with our big dataset, the fitting of the data was so much time-consuming that pull us back from even try a prediction. These methods are: Multinomial Naive Bayes and the Random Forests.

The first one implements the naive Bayes algorithm for multinomially distributed data [7].

Instead, a random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. This tool was a bit faster than the previous one, but however, after few tuning tries, it presents some overfitting problems, due also to the small number of estimators used, kept small in order to avoid the time-consuming issue [8].

4. EXPERIMENTS

After the preliminary exploration of the data, where the goal was to find some significant correlations through a visual and graphic way, the hard work comes. Since the data were provided by the Kaggle platform (www.kaggle.com), we were actually provided of an online service for the model testing. However, the first thing was to split the data into two different datasets.

4.1 Data split

The Kaggle training set has 878049 samples and we split it with a 0.75/0.25 ratio, building the final training set with 658536 labeled samples and a validating set with other 219513 labeled samples. As could have been already understood, the validation approach used is the one with the three different sets of Train, Validation and Test. Indeed, when it was possible, 10-fold Cross Validation was performed in order to tune some parameters.

4.2 Pre-processing phase

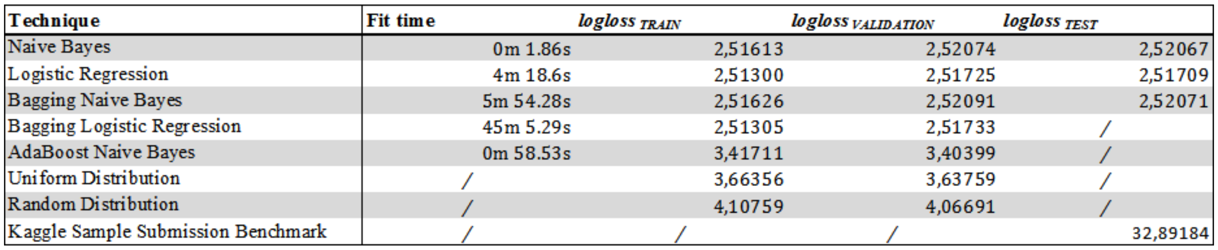
The preprocessing of the data was performed in order to convert categorical variable into indicator variables and to normalize the continuous valued ones. The reasons to do these steps, especially the dummy translation was performed for two reason:

1. Satisfy the model fittings, that often requires no categorical features;
2. Mine most knowledge as possible from these variables, such as the Date field. Indeed, different approaches are available in order to represent a datetime, such as timestamps, or spreading it out in different columns, one for each sub-field of a datetime (Year, Month, Day, Hour, …). I prefer the explosion into indicator variables in order to better catch cyclical components in the crimes trend, that maybe are not so catchable with the absolute values described before.

The last feature engineering step was to exploit the Address field: we simply consider two different categories of Address, the ones with the word ‘Block’ present in the description and the ones with not. In this way we add a new attribute *Block* to each sample, set to 1 in case the word is present, 0 otherwise.

4.3 Fitting the models

The different models were fitted with different features in order to find the best combination of input variables for the final probability estimation. It turned out that fitting the model trained with the features $[c^{Hour}, c^{Day}, c^{Year}, c^{Block}, c^{PdDistrict}, c^{X}, c^{Y}]$ was the best way. From the results provided in the next table (Figure 6), we can notice how our models are not over- or under-fitting the data.



Technique	Fit time	$logloss_{TRAIN}$	$logloss_{VALIDATION}$	$logloss_{TEST}$
Naive Bayes	0m 1.86s	2,51613	2,52074	2,52067
Logistic Regression	4m 18.6s	2,51300	2,51725	2,51709
Bagging Naive Bayes	5m 54.28s	2,51626	2,52091	2,52071
Bagging Logistic Regression	45m 5.29s	2,51305	2,51733	/
AdaBoost Naive Bayes	0m 58.53s	3,41711	3,40399	/
Uniform Distribution	/	3,66356	3,63759	/
Random Distribution	/	4,10759	4,06691	/
Kaggle Sample Submission Benchmark	/	/	/	32,89184

Figure 6: Performances for the different techniques adopted.

In the Uniform Distribution we estimate each crime with equal probability to each possible Category class (therefore, with 39 different Category classes, each estimated probability is $1/39 \cong 0{,}02564$). In the Random Distribution, the probability are totally random.

The strangest thing is how the bagging and the boosting of our models was useless, rather they are sometimes worsening the previous “simple” results. It has been said that bagging typically helps when the base model is overfitting the data or there is high dependency on the training set, and this was not the case. Boosting instead is maybe corrupted by the noisiness of the dataset.

Closing, the two major models adopted (Naïve Bayes classifier for multivariate Bernoulli models and Logistic Regression) performed reasonably well, with very similar performances, improving a lot the sample submission provided by Kaggle and also the Random and the Uniform Distribution.

5. CONCLUSIONS

After a preliminary exploration of the data, mining correlations between the main samples attributes, we have compared two different models, achieving nice and similar results. Maybe we can improve a little more the performances with a sort of clustering of the streets or something similar to better exploit the Address field, and try stronger methods, such as Support Vector Classifiers, that with my machine were not possible to fit, due to the dataset dimensions.

6. REFERENCES

[1] "Multi Class Log Loss," 5 April 2016. [Online]. Available: https://www.kaggle.com/wiki/MultiClassLogLoss.

[2] «Bernoulli Naive Bayes,» 2010 - 2014. [Online]. Available: http://scikit-learn.org/stable/modules/naive_bayes.html#bernoulli-naive-bayes.

[3] «Logistic Regression,» 2010 - 2014. [Online]. Available: http://scikit-learn.org/stable/modules/linear_model.html#logistic-regression.

[4] «Bagging Classifier,» 2010 - 2014. [Online]. Available: http://scikit-learn.org/stable/modules/ensemble.html#bagging.

[5] Y. F. a. R. Schapire, « A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting,» 1997.

[6] «AdaBoost Classifier,» 2010 - 2014. [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html.

[7] «Multinomial Naive Bayes,» 2010 - 2014. [Online]. Available: http://scikit-learn.org/stable/modules/naive_bayes.html#multinomial-naive-bayes.

[8] «Random Forest,» 2010 - 2014. [Online]. Available: http://scikit-learn.org/stable/modules/ensemble.html#random-forests.