Predicting Indicators of Social Inequality with Highway Data

Lynne Burks[†*] **Mahalia Miller**[†*] **Daniel Wiesenthal**[‡*]
Stanford University, Stanford, CA 94305 USA
[†]{lynne.burks, mahalia}@stanford.edu
[‡]dw@cs.stanford.edu

Abstract

Social inequality is an important contemporary issue, and there is high demand for better information about its governing dynamics. Indicators of social inequality may include unemployment, crime rates, number of welfare recipients, and number of homeless people receiving assistance. In this paper, we apply supervised learning methods to predict such socio-economic metrics. We train on data from a large network of highway sensors throughout the San Francisco Bay Area —3,000 sensors gathering car speed and flow in each highway lane every 30 seconds for 10 years. To make use of this large amount of data, we develop a novel machine learning system capable of efficiently working with large datasets using commodity hardware by leveraging a database backend and parallel cloud computing services. We use the machine learning package, Weka, to train classification models. Ultimately, we are able to predict with high accuracy several socio-economic indicators of social inequality using highway traffic data: percent of Californians on welfare, number of Californians on food stamps, unemployment rates, and crime in Oakland.

Introduction

An increasingly relevant and popular metric for the health of a culture or community is the degree to which the population exhibits social inequality. While concept of social inequality is not well-defined, more familiar and concrete measurements of the population–such as unemployment and crime rates, or welfare and food stamp usage–are metrics which, in aggregate, can present a reasonably clear picture of the disparities between different socio-economic strata (Jenkins and Micklewright 2007). Unfortunately, these metrics are sometimes available only after a significant delay, at which point the information is informative but not actionable. If it were possible to access these metrics in real time, it might be feasible to proactively address problems before they fully form–for example, by deploying more patrol cars to locations that are likely to see an increase in crime rates.

Other metrics of a population are much more readily available. For example, the State of California has a network of over 25,000 highway traffic sensors, which gather high volumes of data automatically and have been doing so for the past 10 years. These data, which may seem far removed from issues such as social inequality, may nonetheless capture subtle trends in a community. For example, researchers such as Hadayeghi et al. (2003) have noted correlation of traffic and unemployment.

In this paper, we propose to use machine learning on automatically gathered and readily available data such as highway traffic to predict vital population metrics–such as crime rate and unemployment–in real time, in order to both better understand the health of a population and to provide actionable information about current concerns.

[*]The first three authors contributed equally to this work.

Machine Learning System Design

We built a custom machine learning system from scratch in Python, leveraging MongoDB as a database backend and Amazon Web Services (AWS) for parallel cloud computing. Our Python package offers submodules to create, manage, and evaluate datasets, featuresets (sets of feature extracting functions), and classifiers (see Figure 1 for a flow chart). We take an object-oriented approach, and strive to provide a simple high-level interface:

```
ds = Dataset("our_dataset")
fs = Featureset("our_featureset")
p = Projection(ds, fs, "our_projection")
c = Classifier("NaiveBayes")
c.train(p.projected_datums())
e = Evaluator(c, p.projected_datums())
e.split_dataset(dev_percent=80,
cv_percent_within_dev=25)
e.report_cv_performance()
e.learning_curve(intervals=10, folds=10)
```

We found that the most constraining step in our workflow was the creation of feature vectors from our data. Hence, the primary goal of our system was to make this process fast and easy. We use a metaphor of projecting a dataset through a featureset to create feature vectors; this can all be done in one line of code. The feature projection process takes place in parallel across either all local cores, or, optionally, AWS machines. This is made possible by our database backend (and friendly abstraction layer), which offers simultaneous and indexed access to our dataset, made efficient by carefully tuning our MongoDB indexes. Using this system, we are able to easily and efficiently create feature vectors from our raw data.

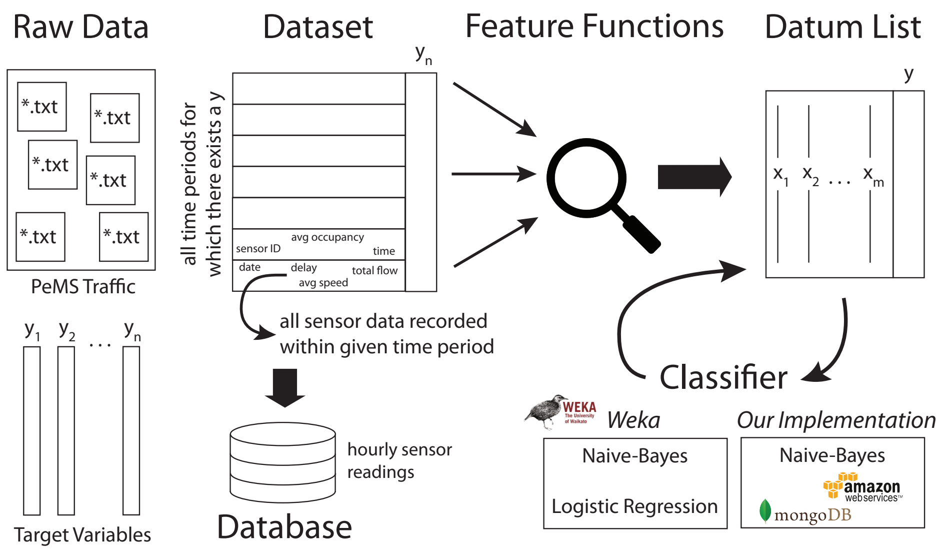

Figure 1: Flow chart of our machine learning Python package

Our package also offers a creative approach to a Naïve Bayes classifier that is particularly well-suited for environments in which new datums are frequently added to the training set. Our approach has three interesting characteristics: instant training, parallel classification, and real-valued feature support.

The classifier stores all datums in its training set in a separate database collection and, when asked to classify an unknown datum, it queries the database holding these training datums to gather the information needed. (For example, to calculate the likelihood that a particular feature has a particular value given some class c, one step is to query for all datums in the training set that have class c.) This means that rather than have a load $\rightarrow$ train $\rightarrow$ classify cycle in which the training phase loops over all datums gathering counts, we have a load $\rightarrow$ classify cycle, in which the counting work is done for us by the database. The result is that we have essentially instantaneous training since the real work is done during classification, and yet the classification step is still relatively efficient, because by leveraging database indexes, we essentially have the database do the bookkeeping for us. In addition to instant training, we can instantly re-train our classifier by simply associating it with a new set of training datums (which may be a super- or sub-set of the original training set); new classifications will immediately take advantage of this additional datum.

Because we do lose some speed when classifying unseen datums by not counting and caching in a training phase and then simply applying the cached counts, we try to make up for it in other ways. One way is through nested parallelization. When given a list of datums to classify, we parallelize at no fewer than three levels: first, we classify each datum in parallel. Second, when classifying each datum, we evaluate the $prior * likelihood$ for each class in parallel. Third, we evaluate the likelihood that a feature has a certain value given the class for each feature in parallel. This high degree of parallelism means that individual datum classifications are relatively slow due to the overhead of spawning so many parallel calls, but when classifying a large number of datums, we are able to use practically limitless CPU power (via AWS), so speed becomes directly dependent on our database performance. We found this to be an interesting characteristic of our classifier; rather than being CPU-bound, our system is IO-bound.

Vanilla Naïve Bayes classifiers do not support real-valued features, which we found to be less than ideal. We therefore built real-valued feature support into our classifier. When calculating the likelihood that a feature has a particular value given some class, we model all observed values with these characteristics as a Gaussian distribution, and calculate the likelihood that a value would fall into a small interval surrounding the particular value of interest.

Because we wished to smooth our data, we extended this Gaussian approach in a way inspired by Laplace smoothing– we increased the σ used in our model by multiplying it by a fixed constant (since adding some k would not respect the differing variations of our feature values). This constant is roughly equivalent to an add-k approach in a traditional Naïve Bayes classifier, as it takes probability mass from the most likely values and spreads it out across less likely values, such that no value gets a probability of 0–because, of course, it's a bad idea to estimate the probability of some event to be 0 just because we haven't seen it in our finite training set.

We found it to not be immediately clear theoretically what should happen if in the training set, we have never seen the feature take on any value (given the class). Empirically, what we settled on was just offering a hand-coded very small likelihood. This corresponds theoretically to arguing that if given some class we have never seen this feature take on any value (perhaps due to sensor malfunction, as happened in our data), then the likelihood that we will see *any* value is very low.

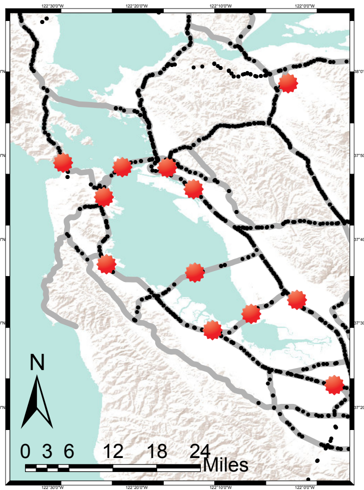

Figure 2: Map of all road sensors in the San Francisco Bay Area, with key locations of interest to this project shown as red dots

Data

Dataset Description

For this project, we used two types of data: highway traffic data as predictor variables and socio-economic data as target variables. The highway data comes from the Caltrans Performance Measurement System (PeMS), which is publically available at pems.dot.ca.gov (Choe, Skabardonis, and Varaiya 2001). PeMS consists of over 25,000 road sensors located on major highways throughout the state of California that have recorded the speed and flow of passing cars in all lanes every 30 seconds for the past decade. We used a version of this data aggregated by the hour in order to make the data more manageable without losing too much granularity. For this project, we focused on sensors located around 15 key locations in the San Francisco Bay Area, including bridges, airports, and metropolitan areas. Figure 2 shows a map of all road sensors in the Bay Area, with chosen key locations marked by red dots.

Choosing measures of inequality is a difficult problem, which has been a topic of open discussion in the field of econometrics and economic theory over the last few decades (Atkinson 1970). Motivated by interest from economics to answer questions regarding poverty and wealth distribution, we selected relevant features related to unemployment and public assistance from different public data sources. Specifically, we examined eight socio-economic factors: (1) BART ridership, (2) number of Californians on food stamps (governmental assistance to buy food), (3) gas prices in California, (4) number of homeless families that are receiving assistance from the State of California, (5) crime rates in Oakland, (6) percent of the California population that is on welfare, (7) crime rates in San Francisco, and (8) unemployment rates in California. Each of these statistics are reported monthly, and Table 1 shows the number of data points in each set with the data source as a footnote.

Table 1: Size of socio-economic datasets

Economic Factor	Dataset Size
BART ridership[1]	12
Food stamps[2]	118
Gas price[3]	118
Homeless[4]	117
Oakland crime[5]	21
Welfare[6]	116
San Francisco crime[7]	43
Unemployment[8]	116

Dataset Preparation

Defining data points for the purpose of classification in this application is a non-trivial task. Our dataset has multiple dimensions, namely space (the sensor recordings were geographically distributed across the San Francisco Bay Area), time, and recording dimensions (flow, occupancy, and velocity for each lane). In addition, the number of data points of the target variables are limited to once per month for the last ten years, so we expect limitations on how many features will be optimal to train our model. Thus, we investigated different choices for defining predictor variables.

We define a data point as corresponding to selected measurements from a given month as well as the corresponding target variable. Since we have nearly a decade of data, we have 118 data points. As previously described, we focus on data from 15 key locations across the San Francisco Bay Area and we aggregate data from about 50 relevant sensors, which is a subsample of our entire dataset, as shown in Figure 2. For each month-long time period, we separately average the flow, speed, and delay recordings during morning and afternoon rush hour for each day of the week at each chosen location. We could easily increase the number of features by including data from more sensors or using flow from every day of the month rather than averaged across days of the week. We could also reduce the number of features by aggregating over an entire month, but we notice that the rush hour time periods and the weekly variation offer more predictive power, perhaps because rush hour dynamics are intuitively more related to socio-economic factors.

The schema for our data points is shown here:

```
<loc1_avgflow_am_Monday,
loc1_avgspeed_am_Monday,
loc1_delay_am_Monday,
...loc15_delay_pm_Sunday, y>
```

[1]http://www.bart.gov/about/reports/ridership.aspx

[2]http://www.dss.cahwnet.gov/research/PG353.htm

[3]http://www.eia.gov/oil_gas/petroleum/data_publications/wrgp/mogas_history.html

[4]http://www.cdss.ca.gov/research/PG283.htm

[5]http://www2.oaklandnet.com/Government/o/OPD/s/CrimeReportArchives/DOWD006169

[6]http://www.cdss.ca.gov/research/PG370.htm

[7]http://sf-police.org/index.aspx?page=734

[8]http://www.labormarketinfo.edd.ca.gov/cgi/dataanalysis/labForceReport.asp?menuchoice=LABFORCE

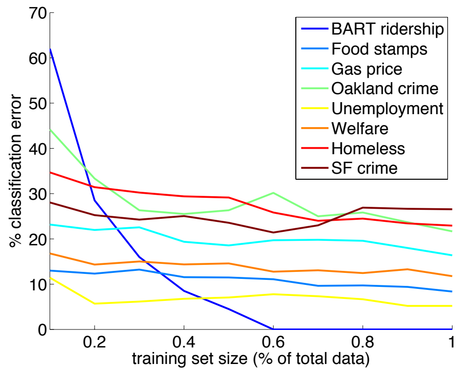

Figure 3: Learning curves of all socio-economic factors, computed using logistic regression in Weka

Our real-valued target variables are each mapped to two classes in order to use common machine learning classification algorithms. We initially tried two different mappings to classes: one corresponding to whether the given point in time had values higher or lower than the average over our whole dataset, and another corresponding to whether the value increased or decreased relative to the previous month. Unfortunately, the first of these mappings gave us unreasonably high predictive performance, and the second gave us very low performance (see discussion in the Results section).

Results

Preliminary Results

As a first pass analysis of our data, we took the feature vectors computed using our machine learning system and analyzed them in Weka with Naïve-Bayes and logistic regression classifiers (Hall et al. 2009). The resulting classification error for each socio-economic factor calculated using both learning algorithms and 10-fold cross validation is shown in Table 2. Based on these results, it appears that highway traffic data is a relatively good predictor of how much the population of California is a) using welfare, b) on food stamps, and c) receiving unemployment assistance.

We also used Weka to compute a learning curve, which shows how the classification error changes as the training size increases (see Figure 3). For most socio-economic factors, the testing error does not change with increased training set size, indicating that obtaining more data will not increase accuracy. Based on this observation, we decided to try improving our models using feature selection.

Feature Selection

We performed feature selection for each socio-economic factor in Weka by computing the information gain for all 630 features, which is a measure of the reduction in uncertainty of the target variable given an observation of the feature of interest. Then for each socio-economic factor, we chose only the features with the largest information gains and retrained our model. For most socio-economic factors, the error is significantly reduced by using this optimized set of features (see Table 2). Classification of BART ridership actually has an error of 0, probably because this dataset only contains 12 data points and is being overfit.

See Table 3 for a list of the most important features for some socio-economic factors. Half of the features that are important for predicting crime in Oakland come directly from sensors located in Oakland or around the Oakland aiport, which makes intuitive sense. For unemployment rates, most of the important features are related to delay over bridges.

Table 2: Classification error using logistic regression and Naïve-Bayes with 10-fold cross validation

	All features		Feature subset	
Indicator	Logistic Reg.	Naïve Bayes	Logistic Reg.	Naïve Bayes
BART ridership	8.3	58.3	0.0	8.3
Food stamps	1.7	9.3	8.5	5.1
Gas price	17.8	17.8	16.9	13.6
Homeless	37.6	30.8	31.4	34.2
Oakland crime	52.4	61.9	23.8	14.3
Welfare	5.2	8.6	12.1	6.0
SF crime	27.9	14.0	23.3	14.0
Unemployment	7.8	6.0	6.9	1.7

Interpretation

Given the incredibly high accuracy in our classification models, we re-examined our choice of target variables. We noted the common dependence on time of some of our predictor and target variables. For example, in the first implementation we separated percent of the population on welfare into two classes: one being below average over the last decade and one being above. However, welfare was above average for 2 years, then below for 4 years, then above until present, showing very limited variability with time (see Figure 4). Similarly, traffic flow in some locations followed a general increase with time. Thus, the high accuracy in classification can be argued to be a trivial result for some socio-economic factors. But we were able to predict crime rates in Oakland with only 14.3% error, even though this data shows a large variability with time as shown in Figure 4.

Because of this common dependence on time, we investigated other choices of determining classes, such as using a moving average with a bin width of 6 months or two years. The resulting model had high classification error near 50%, indicating the model was not better than randomly predicting a class. Future work could investigate the proper choice of classes so as to make the moving average concept meaningful.

Another consideration was to use traffic data from a given month to predict the class of the social indicator for the upcoming month. For target variables that rarely changed from month to month (like welfare), the predictive ability of our models was about the same as classifying social indicators using traffic data from the same month. However, for crime

Table 3: Features with highest information gain for crime rates in Oakland and unemployment

Oakland crime		Unemployment	
Info Gain	Feature Name	Info Gain	Feature Name
0.615	OAK_AIRPORT_AVGSPEED_AM_FRI	0.810	BRIDGES_AVGDELAY_PM_SAT
0.499	PALOALTO_AVGFLOW_AM_THU	0.653	BRIDGES_AVGDELAY_AM_TUE
0.411	OAK_AVGFLOW_PM_WED	0.650	BRIDGES_AVGDELAY_PM_THU
0.401	SANJOSE_AVGSPEED_PM_WED	0.618	SF_AVGDELAY_AM_THU
0.401	OAK_AIRPORT_AVGDELAY_AM_WED	0.614	OAK_AVGSPEED_PM_SAT
0.401	SANTAROSA_AVGSPEED_PM_TUE	0.586	BRIDGES_AVGDELAY_PM_SUN
0.401	OAK_AVGSPEED_PM_SAT	0.505	OAK_AIRPORT_AVGSPEED_PM_SAT
0.342	SANJOSE_AVGDELAY_AM_SUN	0.501	OAK_AIRPORT_AVGSPEED_PM_MON
0.342	OAK_AIRPORT_AVGSPEED_PM_TUE	0.470	FREMONT_AVGDELAY_PM_MON
0.240	PALOALTO_AVGSPEED_AM_THU	0.467	SANJOSE_AVGFLOW_AM_MON

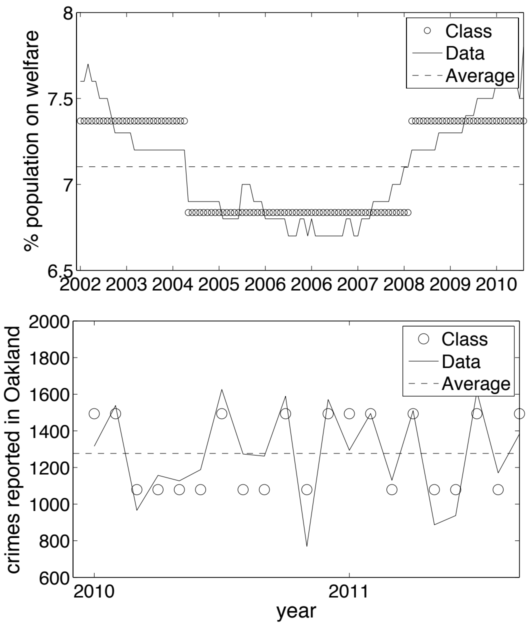

Figure 4: Inequality indicator vs. time, shown as both actual continuous data and our classification of above or below average where the indicator is percentage of Californians on welfare and crime in Oakland.

in Oakland, for example, we had 30% classification error predicting the future month as compared to 15% when predicting within the same month.

Conclusions

In this paper, we train classification models to predict social inequality like the percent of Californians on welfare, the number of people in CA receiving food stamps, and crime reports per month. We build the set of features from a large dataset of highway sensor recordings using our custom machine learning system. We apply this framework to historical data from highway sensors in the San Francisco Bay Area and indicators of local social inequality over the last decade.

Common dependence of our predictor and target variables, such as on time, poses challenges for gaining non-trivial results, especially for unemployment rates, number of Californians on food stamps, and percent of the population on welfare. Thus, we discuss appropriate choices for binning target variables.

We show that recordings from highway sensors are good predictors of monthly reports of crime in Oakland, CA and warrant further investigation.

Acknowledgments

We would like to thank Andrew Ng for insightful discussions and Stanford University for resources.

References

Atkinson, A. 1970. On the measurement of inequality. *Journal of Economic Theory* 2:244–263.

Choe, T.; Skabardonis, A.; and Varaiya, P. 2001. Freeway performance measurement system (PeMS): an operational analyis tool. *Transportation Research Board Annual Meeting*.

Hadayeghi, A.; Shalaby, A.; and Persaud, B. 2003. Macrolevel accident prediction models for evaluating safety of urban transportation systems. *Transportation Research Record* 1840(1):87–95.

Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; and Witten, I. H. 2009. The WEKA data mining software. *ACM SIGKDD Explorations Newsletter* 11(1):10.

Helpman, E.; Itskhoki, O.; and Redding, S. 2010. Inequality and unemployment in a global economy. *Econometrica* 78(4):1239–1283.

Jenkins, S., and Micklewright, J. 2007. *Inequality and Poverty Re-examined.* Oxford, UK: Oxford University Press.

Khattak, A. J.; Wang, X.; and Zhang, H. 2010. Spatial analysis and modeling of traffic incidents for proactive incident management and strategic planning. *Transportation Research Record: Journal of the Transportation Research Board* 2178(-1):128–137.