San Francisco Crime Classification and Prediction

Kai Wang
Department of Computer Science, UCLA
kwang42@cs.ucla.edu

Abstract—**In this paper, we apply machine learning and time series analysis to the problem of classifying and forecasting crime incidents in San Francisco. Our dataset originates from a Kaggle competition [1]. Based on existing researches on these problems, we employ Logistic Regression and VAR(p) models respectively. The classification problem result demonstrates high challenge as our results across all the 39 crime categories achieved the accuracy of 32.67%. For the time series analysis, we revealed correlations of occurrences of difference crime categories. And the VAR(p) model can forecast the trend of number of certain crimes. We also involved demographic data for classification problem, which contributed to a slight improvement to accuracy. With richer and more up-to-date data, our results might be further improved.**

Keywords: Crime Prediction; Feature Engineering; Classification; Time Series Analysis

I. Introduction

Crime activities have always been threat to public safety, and researches on criminals have long been limited within the realm social science. With the huge volume of existing crime datasets and the advent of robust machine learning and statistical technologies, we seek to predictively analyze crime incidents. Our vision is to help police, public policy makers as well as individuals understand crime patterns and take more effective measures to prevent crimes.

In our project, we combined spatio-temporal and demographic data to predict which category of crime has the largest possibility to occur, given a timestamp, location and demographics of the surrounding area. The inputs to our classification algorithm include time (hour, day, year), location(street, latitude, longitude and police district) and demographic data(population density, median house value, poverty rate and median per capita income). The output is the type of crime that is most likely to have occurred. We performed complicated feature engineering to enrich predictors. We also tried various classification algorithms, among which Logistic Regression achieved highest accuracy.

For time series analysis, we applied vector autogression model to capture the interdependencies among multiple categories of crime time series. We tested for the optimal lag p for VAR(p) model. The input to VAR(p) is spatio-temporal data; the output is the time series model. For some crime categories, this model can reveal the trends of its occurrence.

II. Datasets

Our primary dataset is a training set obtained from Kaggle, which includes 878,049 crimes in San francisco that occurred from 2003 to 2015. Every crime record was marked a category, and there are in total 39 categories. The other dataset is the demographic data of different neighborhoods in San Francisco, including population, income, race constitution, education level etc. The demographic data is provided by US Census conducted in 2010. We also searched Wikipedia for population density.

A. Features

Each record in our dataset correspond to a particular crime incident, which includes following fields:

- Date and timestamp of the crime
- Category of the crime. Which is the label to be predicted in test data set.
- Detailed description of the crime (only provided in train data)
- Day of the week of the incident
- Name of the Police Department District the took charge of the crime.
- Resolution, which means how the crime was resolved(booked, arrested or resolved, etc)
- Approximate street address of the crime
- Longitude
- Latitude

Among these fields, resolution is discarded as it only exist in train data.

B. Preprocessing

While http://datasf.org also includes similar data, our dataset from Kaggle is more selected as there is no empty cell in any fields. However, some fields contain unreasonable values due to probably wrong registration. Some numeric fields need to be scaled while some fields are not yet numeric. To transform the original data to suitable input of classification algorithms, we performed feature engineering described below.

1) Outlier removal: The following pictures in Fig.1 are scatter plots of scaled longitude(X) and latitude(Y). There are in total 67 outlier points, all of their latitudes are beyond the border of San Francisco. After removal of these outliers, the scatter plot fits the territory of San Francisco.

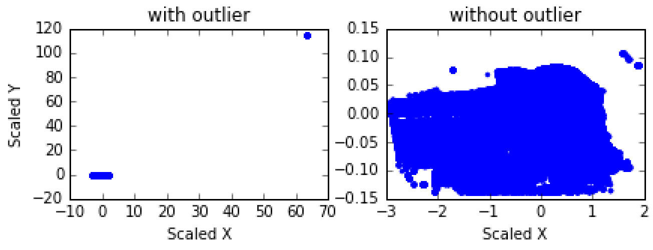

Fig. 1. Scatter plot of crime locations before and after outlier removal

2) Demographic data: Our demographic data is based on US Census from 2006 to 2010. It includes demographic data in each neighborhood of San Francisco, covering race, education, income and housing characteristics. The heatmap in Fig 2 displayed the correlation among the frequency of some typical crime categories and demographic statistics. For example, poverty rate is positively correlated to most crime categories, especially embezzlement; while high population density tend to relate to drug use and disorderly conduct.

One challenge of employing demographic data is correctly matching it the the crime location as the crime dataset does not include neighborhood. Our solution was to download the map data of San Francisco, where each neighborhood is represented as a polygon. Following pseudo-code shows how we located each neighborhood.

```
bool getNeighborhood(Lang, Lat, Polygon){
  //For acceleration
  if (Lang, Lat) in Polygon-Bounding-Box{
    if (Lang, Lat) in Polygon{
      return True
    }
  }
  return False
}
```

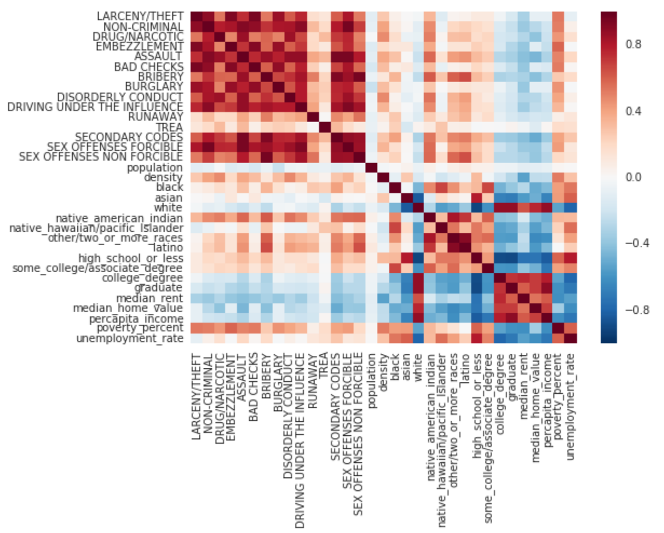

Fig. 2. heat map of correlation among crime amount and demographic data

3) Feature enrichment: For timestamps in dataset, we parsed the monthdate-of-year, hour-of-day from them. We also added season, day/night as new fields.

Another feature enrichment approach is computing the log-odds defined below, where c is crime category.

$$logodds\_c = log(\frac{c\_counts}{len(train\_data)}) - log(1 - \frac{c\_counts}{len(train\_data)})$$

There are around 23,000 addresses in the train dataset, which is small compared with number of crime entries.

4) Scaling and Transformation: Some fields are skewed, with log transformation their distribution are more close to Gaussian Distribution. Figure 3 shows the log transformation of percapita-income. Similar transformations are performed on median-rent and population density.

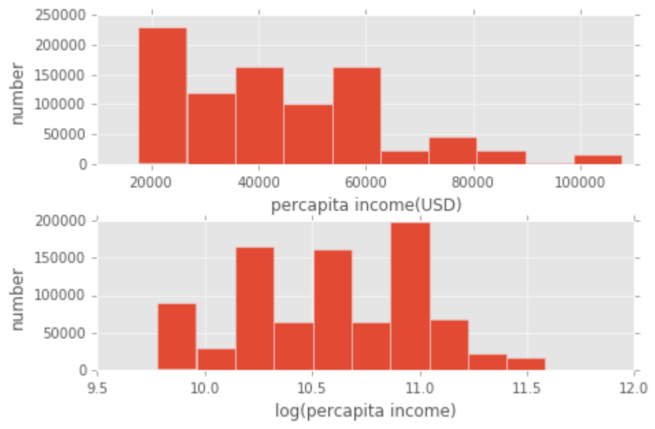

Fig. 3.

III. Methods

A. Classification Algorithms

Our choice of classification algorithm considers number of training samples, independence among features, linear separability as well as overfitting problem. Among various classifiers, Support Vector Machine(SVM) and K-Nearest Neighbors(KNN) are first abandoned because of their low computational efficiency. Our dataset has hundreds of thousands of entries and these algorithms would run for hours. Following are the algorithms selected for our classification experiments.

1) Logistic Regression: Logistic regression is a pretty well-bahaved classification algorithm that can be applied as long as predictors are roughly linear. We assume that our preprocessing have constructed a linear separable feature set and properly scaled numeric features. It is fast, and less vulnerable to overfitting by applying a simple 'l2' penalty function.

Logistic regression employs a 'logit' model:

$$ln\frac{p}{1-p} = \mathbf{a} + \mathbf{B}X + e$$

where p is probability that event $Y = \mathbf{a} + \mathbf{B}X + e$ happens. The fitting goal is to make likelihood

$$L = Prob(p_1 * p_2 * ...p_n)$$

as large as possible by finding the coefficients $(\mathbf{a}, \mathbf{B})$. The log of L, namely log likelihood is usually calculated instead of L itself, since $log()$ is monotonic. The python sklearn library implemented logistic regression, and we use its result as the baseline of other classification experiments.

2) Random Forest: Random Forest is a variation of CART(Classification and regression trees) models. CART models are popular for several advantages. They are easy to interpret, as we can derive series of rules by post-processing the tree. They can handle mixed discrete and continuous inputs, which is a characteristic of our spatio-temporal dataset. They are also able perform automatic variable selection, and are relatively robust to outliers [2].

However, CART models have evident disadvantages. They do not predict very accurately due to the greedy nature of the true construction algorithm. Also trees are unstable (also referred to as having high variance), because slight changes at input data could be escalated significantly as the tree grow. One approach to reduce the variance of an estimator is aggregating together many estimates. We can train M different trees on different randomly chosen subset of the data, and then compute the ensemble

$$f(x) = \sum_{m=1}^{M} \frac{1}{M} f_m(x)$$

where f_m is the m'th tree. This technique is called bagging. While running the sample algorithm on subsets could result in highly correlated predictors, the base leaners can be de-correlated by randomly choosing subsets of training data. And this random-choosing regime is referred to as **random forest**. To improve its performance, Bayesian adaptive regression trees are often applied to performing Bayesian space ensembles.

3) Gradient Boost Tree: Boosting is a greedy algorithm for fitting adaptive basis-function models. Gradient Boosting derives a generic loss function:

$$\hat{f} = \underset{f}{\text{argmin}} L(f)$$

where $f = (f(x_1), ..., f(x_N))$ are the weak learners. This is solved with gradient decent stage by stage. At step m, let g_m be the gradient of $L(f)$ evaluated at $f = f_{m-1}$:

$$g_{im} = [\frac{\Delta L(y_i, f(x_i))}{\Delta f(x_i)}]_{f=f_{m-1}}$$

Then the update is made:

$$\mathbf{f}_m = \mathbf{f}_{m-1} - \rho_m \mathbf{g}_m$$

where ρ_m is the step length. The algorithm can be fitted to a weak learner to approximate the negative gradient signal. Hence we update

$$\gamma_m = \underset{\gamma}{\text{argmin}} \sum_{i=1}^{N} (-g_{im} - \phi(\mathbf{x}_i; \gamma))^2$$

The overall algorithm can be applied to log-loss or other loss functions such as Huber loss, which is more robust to outliers. The gradient boost algorithm could also over fit as there are no penalty functions like in Logistic Regression. It takes efforts to tune it for satisfactory performance.

B. Time Series

A time series is a sequence of data points with continuous time interval and successive measurements made over a time interval. Time series analysis comprises approaches for extracting meaningful statistics and other characteristics of the data. And time series forecasting is using a model to predict future values based on previously recorded data. It has been applied to crime analysis for decades [3].

VAR models (vector autoregressive models) are desiged for multivariate time series. It view each variable as a linear combination of past lags of itself and past lags of other variables. The variables(number of crimes in our case) are collected in a 39*1 vector. At time t, define the vector as y_t, a p-th order VAR, namely VAR(p), is

$$y_t = c + A_1 y_{t-1} + A_2 y_{t-2} + ... A_p y_{t-p} + e_t$$

where y_{t-j} is called the $j-th$ lag of y, c is constant intercept and e_t is an error term. The VAR model assumes the data is **weakly stationary** and above attributes of e_t must hold

1) $E(e_t) = 0$, zero mean error
2) $E(e_t e_t') = \Omega$, the contemporaneous covariance matrix of e_t is positive semidefinite.
3) $E(e_t e_{t-k}') = 0$, for any positive k. There is no series correlation in error term.

An Augmented Dickey-Fuller unit root test [4] is often employ to test stationarity. Using the library from statsmodel [5] , we performed the ADF test on each category with constant, linear and quadratic trend. We chose BIC(Bayesian Information Criterion) for penalty strictness. (results in Fig 4). There are 23 categories with scores below 5% level, and we selected these categories for time series model construction.

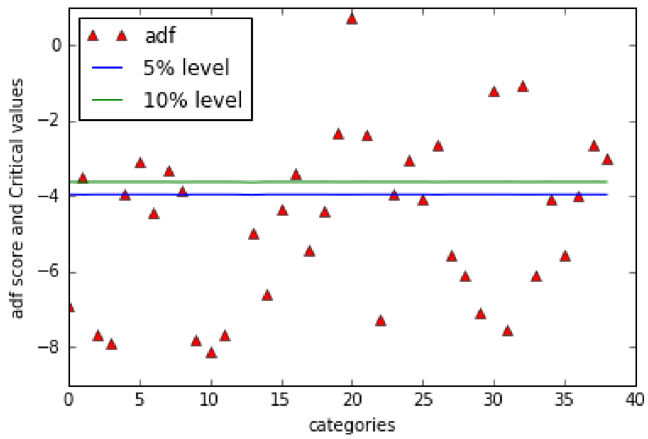

Fig. 4. ADF test score against 5% and 10% threshold

IV. Experiments

A. Classification

We have 878049 labeled spatio-temporal crime records, of which half are trained as training dataset, the rest are treated as test data to verify the accuracy of our classifiers. For each model, there are two kinds of prediction results. One is binomial labels(0 or 1) for each category, the other is possibility of each category. Following table displays the mean accuracy and log loss of each classification algorithms. For Random Forest and Gradient Boost Tree, we put the best results with highest accuracy and lowest log loss.

We tried to incorporate demographic data to enrich our feature set, as introduced in the Datasets chapter. There are 18 demographic fields. To identify their influence, we constructed a matrix whose columns are demographic data and number of each category of crime, rows are neighborhood names. We computed its correlation matrix *matcorr*. For each demographic field in *matcorr*, we construct an array *arrcorr* of its correlation with frequencies of all categories and compute the variance of *arrcorr*. The larger variance it has, the stronger influence this demographic field has on crime category. Following table list eight most influential entries. Apart from race, we see poverty percent, median home value and population density are also prominently informative.

TABLE I
INFLUENTIAL DEMOGRAPHIC DATA

demographic feature	variance
latino ratio	0.0282
native_hawaiian/pacific_Islander ratio	0.0250
other/two_or_more_races ratio	0.0243
black ratio	0.0222
poverty_percent	0.0168
median_home_value	0.0147
population density	0.0143

TABLE II
ACCURACY AND LOG LOSS OF FITTING RESULT

	accuracy	log loss
Logistic Regression	0.3258	2.2176
Random Forest	0.3376	2.2038
Gradient BoostTree	0.3356	2.1866

The table demonstrate that Random Forest yields the highest accuracy and Gradient BoostTree has lowest log loss. Their results did not vary much, even the plain logistic regression also reached an accuracy of 0.3258. There are in total 39 classes, the accuracy of 0.33 is actually not bad. We computed the predict-probability on the test dataset from Kaggle, which has 884263 crime unlabeled crime records. By the time of submission, we ranked 47th on the leaderboard https://www.kaggle.com/c/sf-crime/leaderboard among over 1200 submissions.

1) Tuning Random forest: Our main efforts with Random Forest was to tun the hyper-parameters to mitigate overfitting. The most relevant parameter to this task was the maximum depth of the decision trees(weak classifiers) of which the Random Forest is composed. Besides, the number of estimators could also affect fitting results.

Using Sklearn in Python, We first tried different number of estimators. Fig 5 plots accuracy and log loss as $scaled\_log\_loss = 0.3 + \frac{log\_loss}{100}$. As is shown in the picture, the mean accuracy increases and log-loss reduces as the estimator number increases from 10 to 60, but converges after that.

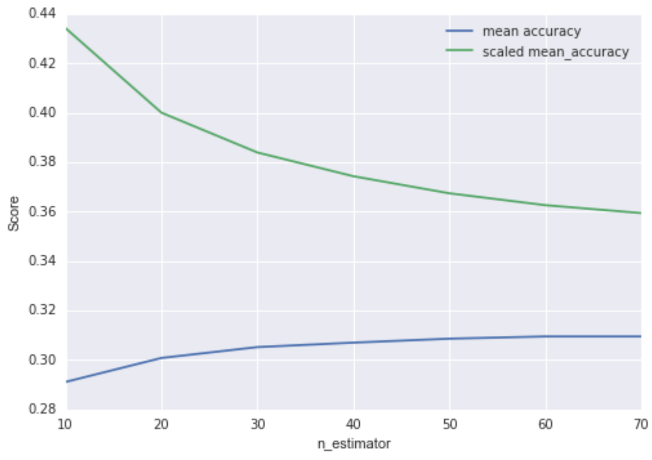

Fig. 5. Mean accuracy and scaled log loss

Considering time efficiency, we took 60 as the number of estimator for further training. To continue tuning the Random Forest algorithm and find its optimal max decision tree depth, we tried different max-depth with same number of estimator. Fig 6 shows the result. When max tree depth is 16, we gain the lowest log loss and highest accuracy.

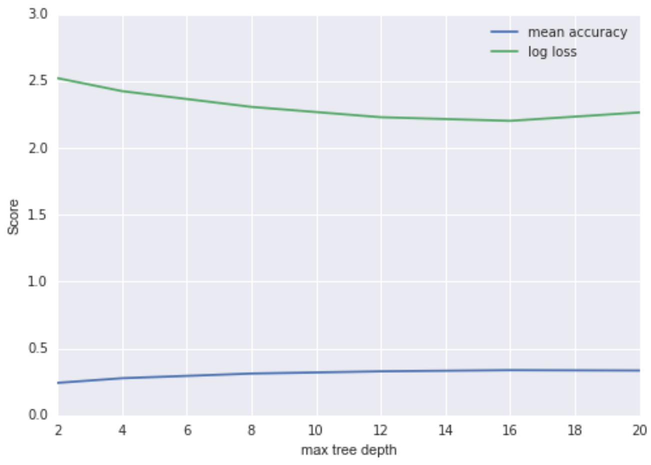

Fig. 6. Mean accuracy and scaled log loss

2) Tuning GRBT: The GRBT is supposed to have superior performance than Random Forest, but is painful to tune the parameters for optimal outcome. Generally, it is better to employ more decision trees for higher prediction accuracy and lower log loss. But our experiment results in fig 7 show that the best result occurs when the n_estimator equals to 50, and increasing it does not improve the results.

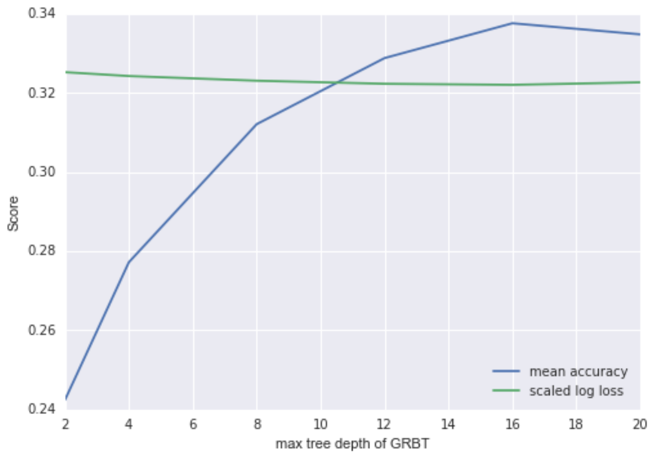

Fig. 7. Mean accuracy and scaled log loss

We also compared the importance of different features with the 'gini importance' and selected the 20 most important predictors as shown in fig 8. **time**(hour of day) turn out to be the most important feature. The logodds account for the majority of influential features. Year also

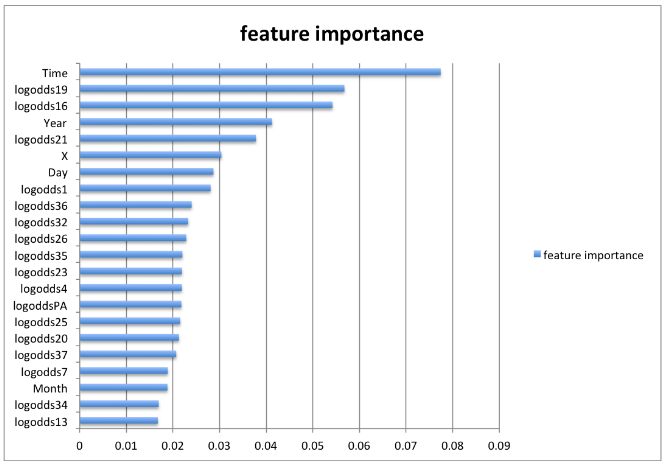

Fig. 8. Mean accuracy and scaled log loss

B. Time Series

1) Motivations: Based on the stationarity test in previous chapter, the VARp model is suitable for finding trends in the crime dataset. We selected 28 categories with low ADF score, and computed the pair-wise correlation among them. Fig 9 shows the heat-map of correlation matrix which computes the Pearson's r. In the heat-map, we can find large positively correlation coefficients, for example, BURGLARY & ASSAULT, as well as negative coefficients like DRIVING UNDER THE INFLUENCE & VEHICLE THEFT.

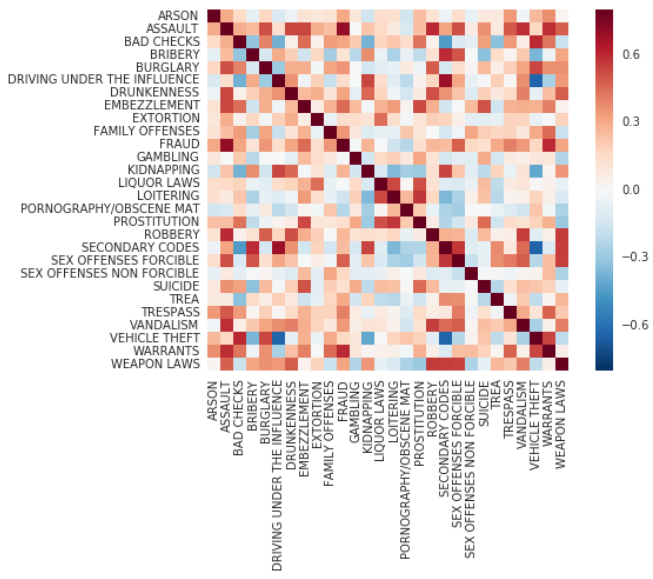

Fig. 9. Heatmap of time series correlation matrix

Auto-correlation function(ACF) and Cross-correlation function(CCF) are also effective approaches to reveal statistical connections. Fig 10 demonstrate the CCF between category 'DRIVING UNDER THE INFLUENCE' and 'VEHICLE THEFT'. The increase of the first category precedes the decrease of the other. These kinds of correlations are further exploited in the correlation matrix.

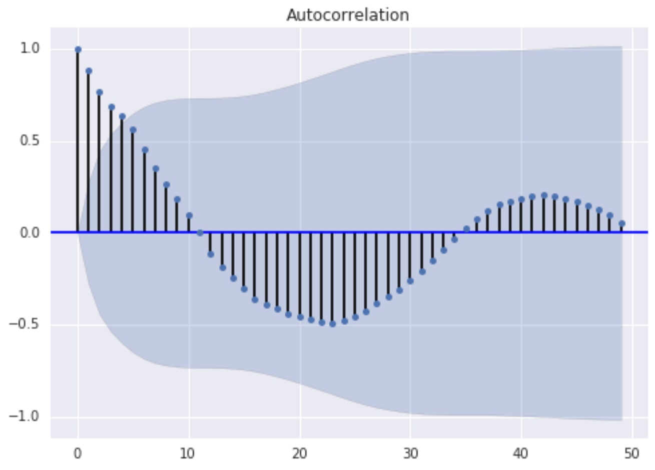

Fig. 10. CCF between 'DRIVING UNDER THE INFLUENCE' and 'VEHICLE THEFT'

2) Model Selection: Choosing the order p for VAR(p) model is a feature selection problem. We tried different values of p under two penalty criterions AIC(Akaike Information Criterion) and BIC. As is shown in Fig 11, with AIC, the penalty score is minimal when $p = 2$. For BIC, the penalty is 30.88 at $p = 1$ and 30.89 at $p = 2$.

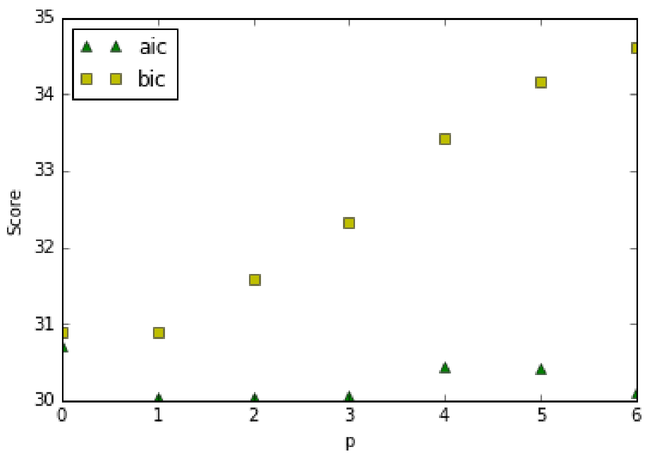

Fig. 11. aic/bic

3) Fitting Model: We choose $p = 2$ and AIC penalty criterion for VAR(p) model fitting. And tried the model on "ASSAULT" time series. The first 40 time units were

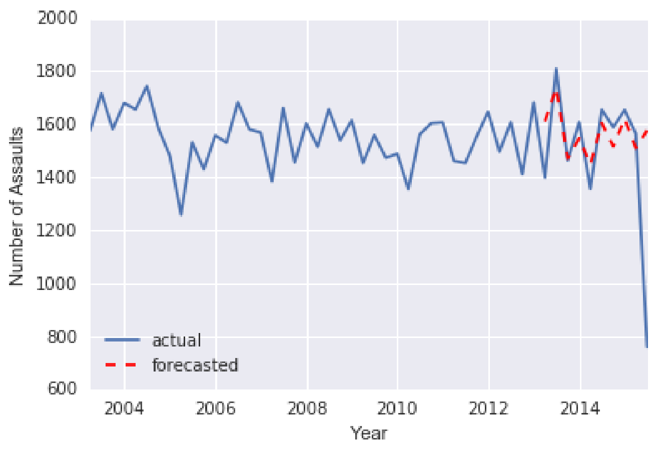

Fig. 12. Forecast of 'ASSAULT'

used for fitting, and the last 10 were forecasted. As Fig 12 shows, the model is not able to capture the exact quarterly variation, but it predicts the trend of increase or decrease in crime number correctly.

V. Related Work

VI. Conclusion

In this paper, we worked on multi-class prediction and time series analysus on crime category in San Francisco. The dataset we employed included labeled spatio-temporal data from Kaggle and demographic data from US Census in 2010.

The prediction task achieved mean accuracy of around 33 percent, which ranked 47th among over 1200 submissions in Kaggle leaderboard. We compared Logistic regression, Random Forest and Gradient BoostTree algorithms. The ensemble tree methods are tuned with different max-tree-depth and number-of-estimators. The best fitting result was from Random Forest, while GRBT still possess the potential to excel other classifiers should it be further tuned with fine granularity. We involved demographic data to the classification task; however, it did not significantly improve the result.

We performed time series analysis on the spatio-temporal data with vector autoregression model. We found the optimal $p-lag$ with AIC and BIC penalty functions. From the heat-map, the crime category time series data also reveal correlation between different categories. The ADF test demonstrated that 23 out of 39 categories have more than 95% probability to be stationary and we fitted VAR(p) model on these categories. The VAR(p) model is also able to predict trends of category number changes.

VII. Ackknowlegement

I would like to deeply thank Professor D. Stott Parker for his advice on feature selection and preprocessing, which facilitated my preprocessing with demographic data.

References

[1] https://www.kaggle.com/c/sf-crime
[2] Murphy, Kevin P. Machine learning: a probabilistic perspective. MIT press, 2012.
[3] Greenberg, David F. "Time series analysis of crime rates." Journal of Quantitative Criminology 17.4 (2001): 291-327.
[4] Said, Said E., and David A. Dickey. "Testing for unit roots in autoregressive-moving average models of unknown order." Biometrika 71.3 (1984): 599-607.
[5] Seabold, Skipper, and Josef Perktold. "Statsmodels: Econometric and statistical modeling with python." Proceedings of the 9th Python in Science Conference. 2010.