PROJECT REPORT

Semester VIII

Predictive Policing

A study on Classification Algorithms to
Predict Crime Status

Under the guidance of:

Dr. Shirshu Verma

IIIT Allahabad

Submitted by:

Niketan Santosh Rane

[IEC2013094]

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the Bachelor of Technology project report entitled **"PREDICTIVE POLICING - A Study on Classification Algorithms to Predict Crime Status"**, being submitted as a part of Project Evaluation to the Department of Information Technology of Indian Institute Of Information Technology, Allahabad, is an authenticated record of my original work under the guidance and supervision of **Dr. Shirshu Verma** from January 2017 to May 2017. I have adequately cited and referenced the original sources and have adhered to all principles of academic honesty and integrity.

Date: 30.04.2017
Place: IIIT Allahabad

Signature

Niketan Santosh Rane
[IEC2013094]

CERTIFICATE FROM SUPERVISOR

This is to certify that the statement made by candidate is correct to the best of my knowledge and belief. The project entitled **"PREDICTIVE POLICING - A Study on Classification Algorithms to Predict Crime Status"** is a record of candidate's work carried out by him under my guidance and supervision. I do hereby declare that it should be accepted in the fulfillment of the requirements of the project of IIIT Allahabad.

Dr. Shirshu Verma
Professor, Dept. of IT,
IIIT Allahabad.
Date: 30.04.2017

ACKNOWLEDGEMENT

I would like to acknowledge and extend my heartfelt gratitude to Dr. Shirshu Verma, Indian Institute of Information Technology Allahabad, who guided me through this project. His keen vital encouragement, superb guidance, and constant support are the motive force behind this project work. I would like to show my warm thanks to all our friends for their continuous motivation and encouragement during this project. I am very thankful to all the technical and non- technical staffs of the college for their assistance and cooperation.

Niketan Santosh Rane
[IEC2013094]

ABSTRACT

In the past few years, there has been a huge increase in crime rate all over the world. The task of crime detection and prevention takes on significant importance in such a scenario. Data driven decisions tend to be more accurate, involve less guesswork, and can be justified more easily. These characteristics make it ideal for sensitive issues like trying to discover underlying patterns in crime data which demand transparency and accountability. We experiment with classification learning algorithms to predict the crime status on a real dataset. We discuss two new algorithms for the same and prove both the effectiveness of data mining in prediction and detection of crime patterns in general. The scope of this project is to prove how effective and accurate the machine learning algorithms used in data mining analysis can be to predict and detect crime patterns.

TABLE OF CONTENTS

1. INTRODUCTION

Criminal activity is an inevitable part of urban life. In the past decade, there has been significant growth in illicit trafficking of drugs, murders, theft and other crime activities. This growth has forced governments to use modern technologies and methods to control and prevent crimes. Since the manual interpretations of crime data are limited due to incredible data size, the raw unreadable format of most data entry, as well the complex interdependencies in the determinants of crime (or, as we will refer to them in future, crime attributes), use of data mining and machine learning tools to identify patterns for detecting future criminal behaviors is effective and essential. Data mining and machine learning methods accelerate crime analytics; provide better analysis and real time solutions thereby saving considerable resources and time.

Many important questions in public safety and protection relate to crime, and a thorough understanding of criminal activities is beneficial in many ways: it can lead to targeted and sensitive practices by police department to alleviate crime. With the availability of fast, efficient algorithms for data analytics, crime detection and prevention is an active and growing field of research.

In this project, a real crime dataset from UCI Machine Learning Laboratory is used. **Voted-Perceptron Algorithm and SMO Algorithm have been implemented** to classify a crime dataset based on a binomial class, the crime status. Then the results are compared with another four different classifiers (Naïve Bayes, Logistic Regression and K-Means), and the most efficient algorithms are identified. These classifiers are popular in many different fields, also in crime, but we will attempt to identify the most effective ones out of the five.

2. Problem Definition

This project primarily aims to compare the different classification algorithms based on accuracy.

The objective of this project is twofold:

1. To analyze **Voted-Perceptron** and **Sequential Minimal Optimization (SMO)** in terms of accuracy when classifying the crime status of a specific neighborhood based on its geographical and socio-economic structure.

2. To show a comparative analysis of proposed algorithms against different classifiers: Naïve Bayes, Logistic Regression, SVM, and K-Means, in terms of AUC so we can reasonably choose more accurate algorithms to classify crime status.

3. Literature Survey

Data mining classification techniques have been widely used for crime forecasting nowadays. Anna L. Buczak in her paper [1], studied the application of fuzzy association rule mining for community crime pattern discovery. This approach helps in finding interesting and meaningful crime patterns of importance to a community.

Melissa Morabito et al. [2] discussed the ensembling of classification algorithms to predict crime. The experimented with different classifiers and different feature sets and used voting method to improve the results.

Brown [3] employed ReCAP, the Regional Crime Analysis Program which helps Charlottesville Police Department to discover patterns and relationship in criminal activities in the area. The System also provide Hot Sheet which gives the summarized criminal activity of the region.

Shyam Varan Nath [4] implemented K-Means clustering algorithm along with geo-spatial plot to detect crime patterns in a particular region. He was one of the first to realize the difference in importance of various factors in predicting crime, i.e. the predictive nature of attributes. He built a weighting scheme for attributes which improved accuracy.

Chen at al. [5] presented a general framework for crime detection and prevention on the basis of the project conducted by the University of Arizona researchers in collaboration with the Tucson and Phoenix police departments.

John C. Platt [6] proposed a novel approach to train a support vector machine. Training a support vector machine requires the solution of a large optimization problem and Platt provided an idea of how to break this problem into sub-problems and solve them analytically using Kuhn Kutcher conditions.

Freund at al. [7] combined Rosenblatt's perceptron algorithm [8] with Vapnik's Classifier Algorithm to classify handwritten digits. They also used kernel based approach to increase the speed of their algorithm considerably.

4. Dataset Description

The dataset used for this project is extracted from UCI Machine Learning Repository and named "Communities and Crime Unnormalized Data Set". The dataset emphasizes on communities of United States, and combines socio-economic data from 1990 US Census, law enforcement data from the 1990 Law Enforcement Management and Administrative Statistics (LEMAS) Survey, and crime data from the 1995 United States FBI Uniform Crime Report. Communities not found in both census and crime datasets were already omitted. The per capita crime variables in the dataset were calculated using population values included in 1995 FBI data.

The dataset consists of 2215 number of instances and 147 attributes. Out of the 147 attributes, 125 attributes are predictive, 4 non-predictive and 18 potential goal attributes. Each instance has a community name which is for information only and not been used for data exploration; which is to say the name of the community has not been used as a determinant in classifying crime since it is an obvious divider which doesn't add to the information gain of the data mining conducted. Further, each instance belong to a unique state which is denoted by 2 letter postal abbreviation code. Other attributes include information across a variety of crime-related features, ranging from percentage of households with minimum wage salary to population density, and percentage of people under poverty level to percentage of police officers per 100K population. Also included are measurements of crime considered violent, such as murder, rape, robbery and assault. The detailed information about the attributes of the dataset can be obtained from the UCI machine learning repository website.

state	population	householdsize	racepctblack	racePctWhite	agePct12t21	numbUrban	medIncome
26	11980	3.1	1.37	91.78	12.47	11980	75122
33	23123	2.82	0.8	95.57	11.01	23123	47917
32	29344	2.43	0.74	94.33	11.36	29344	35669
29	16656	2.4	1.7	97.35	12.55	0	20580
21	140494	2.45	2.51	95.65	18.09	140494	21577
17	28700	2.6	1.6	96.57	11.17	28700	42805
13	59459	2.45	14.2	84.87	15.31	59449	23221
24	74111	2.46	0.35	97.11	16.64	74115	25326
38	103590	2.62	23.14	67.6	19.88	103590	17852
38	31601	2.54	12.63	83.22	15.73	31596	24763
4	25158	2.89	21.34	49.42	13.65	25158	25479
15	40641	2.54	12.18	86.39	21.51	0	20043
2	57140	2.74	53.52	45.65	16.51	57140	19143
6	45532	2.85	2.65	95.72	11.86	43944	44635
4	180038	2.62	1.3	74.02	12.04	180038	34372
30	14869	2.67	2.28	94.74	13.71	14882	49851
38	261721	2.6	8.41	82.64	14.18	261763	35048
29	7322564	2.6	28.71	52.26	13.06	7322564	29823
4	26436	3.34	18.97	53.6	16.16	26436	29043

Figure 1. Training Dataset [clipped]

4.1 Pre-processing:

Data preprocessing is often considered the heart of data mining. An appropriately processed dataset is amenable to most learning algorithms and more easily understood by even laymen. In this project, there are a few methods employed for data preprocessing. The techniques include data cleaning, discretization, data transformation and feature selection. These techniques together reduce noise in data, introduced by human error in manual entry & measurement of values, and incomplete data, also introduced by human error or just plain unavailability or just inapplicability of certain crime attributes to certain crime instances. The processed data is then fed into a classification algorithm. In this project, some communities are removed on the basis of missing values. Certain attributes have more than 80% of missing values, reason being data was not recorded for particular communities. This attributes are removed as they feared to have incomplete data such as *pctPoliceWhite* and *pctPoliceBlack*. The attribute *violentPerPop* has been chosen as a target attribute. Due to this, all the instances where the violentPerPop was missing, had to be removed. This has led to the removal of 221 instances and 1994 instances were remained.

Further, we have implemented min-max normalization technique [0, 1] on all attributes except state in the dataset in order to avoid overflow issue. Then, we categorized the *violentPerPop* (total number of violent crimes per 100K population) into a binomial class "*CrimeStatus*". The threshold value set for the categorization was 20%. The *CrimeStatus* has two values, "Critical (1)" and "Non-Critical (0)." This is done because in order to predict, the target value should be discrete in nature.

5. Proposed Approach

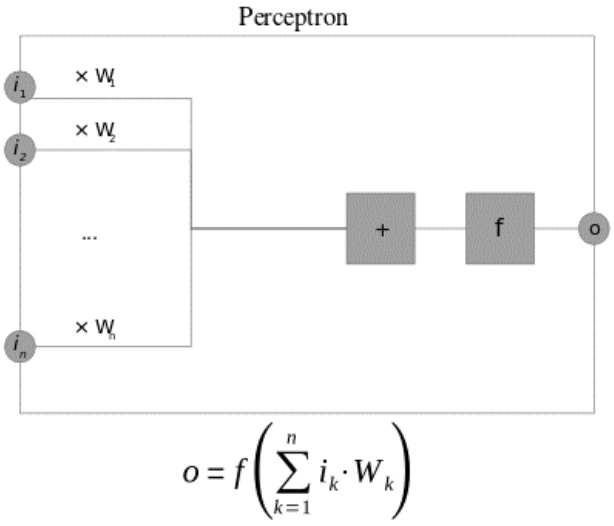
After data preprocessing and feature selection, the number of attributes remaining are now few and meaningful. In this study, we have implemented two new classification algorithms, namely the Voted-**Perceptron** and **Sequential Minimal Optimization** Algorithms. The results of these algorithms are then compared with various classification algorithms, such as Naïve Bayes, Logistic Regression, K-means, and Artificial Neural Networks.

5.1 Voted-Perceptron Algorithm:

The voted-perceptron algorithm is based on the perceptron algorithm of Rosenblatt and Frank. [7] The algorithm is much more efficient in terms of computation time as compared to Support Vector Machines (SVM). The algorithm can be explained as follows:

Suppose we have m training examples. The training data is a matrix with m rows and n columns. Each training example $x \in \mathbb{R}^n$ and represented by values of n different features. Let feature value j for example number i be written as x_{ij}. The label of i[th] example be y_i. Here, in our case y_i = 1 if the *CrimeStatus* is 'critical' and -1 if it is 'non-critical'.

The simplest way to distinguish two classes in n-dimensional Euclidean space $\mathbb{R}^p$ is a hyperplane of dimension $p - 1$. The parameters defining a hyperplane are a vector v in $\mathbb{R}^p$ and a scalar b. The former gives the orientation of the hyperplane, which is at right angles (also called perpendicular, also called orthogonal) to v. The scalar b specifies the distance from the origin to the hyperplane along the direction specified by v.



$$o = f\left(\sum_{k=1}^{n} i_k \cdot W_k\right)$$

Figure 2. A Perceptron algorithm block illustration

The perceptron algorithm is initialized with a zero prediction vector $v = 0$. It predicts the label of new instance x to be

$$\hat{y} = sign(v.x).$$

If this prediction differs from the label y, it updates the prediction vector

$$v = v + y.x$$

If the prediction is correct then the v is not changed. The process then iterates with subsequent examples till convergence.

The most common way the perceptron algorithm learns is to run indefinitely until it finds a prediction vector which is correct on all, or most of its, training set.

In the voted-perceptron algorithm, we store the list of all prediction vectors as intermediate hypotheses. For each prediction vector, we count the number of times it "survives" as survival time c0, c1 … until a mistake is made; which we refer to as weight of the prediction vector. To calculate the prediction, we compute the prediction of all such prediction vectors and combine all of these predictions by a weighted vote. The voted-perceptron after some T iterations over the training set will converge on some consistent hypothesis which will eventually dominate the weighted vote in the algorithm.

5.1.1 Algorithm:

Training

Input: a labeled training set $\langle(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\rangle$
number of epochs T
Output: a list of weighted perceptrons $\langle(\mathbf{v}_1, c_1), \ldots, (\mathbf{v}_k, c_k)\rangle$

- Initialize: $k := 0$, $\mathbf{v}_1 := \mathbf{0}$, $c_1 := 0$.
- Repeat T times:
  - For $i = 1, \ldots, m$:
    - Compute prediction: $\hat{y} := \text{sign}(\mathbf{v}_k \cdot \mathbf{x}_i)$
    - If $\hat{y} = y$ then $c_k := c_k + 1$.
      else $\mathbf{v}_{k+1} := \mathbf{v}_k + y_i \mathbf{x}_i$;
      $c_{k+1} := 1$;
      $k := k + 1$.

Prediction

Given: the list of weighted perceptrons: $\langle(\mathbf{v}_1, c_1), \ldots, (\mathbf{v}_k, c_k)\rangle$
an unlabeled instance: $\mathbf{x}$

compute a predicted label $\hat{y}$ as follows:

$$s = \sum_{i=1}^{k} c_i \, \text{sign}(\mathbf{v}_i \cdot \mathbf{x}); \quad \hat{y} = \text{sign}(s) \; .$$

5.2 Sequential Minimal Optimization:

Sequential minimal optimization is an algorithm used to solve quadratic programming problem that arises during the training of support vector machines. It is an iterative algorithm used for solving a binary classification problem with a dataset $(x_1, y_1) \ldots (x_m, y_m)$, where $x_i \in \mathbb{R}^n$ and $y_i \in \{-1, +1\}$.

Suppose H is a hyperplane separating two classes. The points that lie on hyperplane satisfy $wx + b = 0$, where w is a normal to hyperplane, $\frac{|b|}{\|w\|}$ is the perpendicular distance of the hyperplane to origin. Let d_+ and d_- be the shortest distance from separating hyperplane to positive (negative) example. Define 'margin' to be $d_+ + d_-$. Support vector algorithm simply looks for the separating hyperplane with largest margin. Since the margin is of the form $\frac{constant}{\|w\|}$, minimizing $\|w\|^2$, leads to maximizing the margins subject to constraints:

$$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \quad \forall i \tag{1}$$

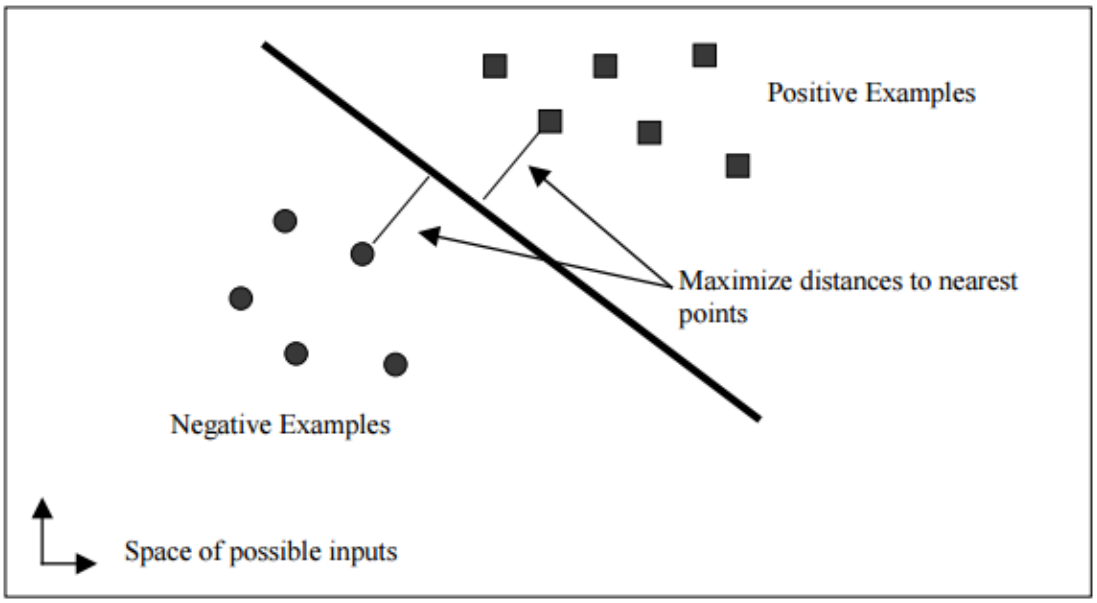


Figure 3. A linear support vector machine

Using a Lagrangian, this optimization problem can be converted into dual form which gives:

$$L_P \equiv \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} \alpha_i y_i(\mathbf{x}_i \cdot \mathbf{w} + b) + \sum_{i=1}^{l} \alpha_i \tag{2}$$

Requiring that the gradient of LP with respect to w and b vanish and substituting the value of *w* in the above equation leads to the following optimization problem:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j K(x_i, x_j) \alpha_i \alpha_j,$$

subject to:

$$0 \leq \alpha_i \leq C, \quad \text{for } i = 1, 2, \ldots, n,$$

$$\sum_{i=1}^{n} y_i \alpha_i = 0 \qquad (3)$$

, where *C* is parameter corresponding to assigning penalty to errors,
$K(x_i, x_j)$ is the kernel function, both supplied by the user:
and the variables α_i are Lagrange multipliers.

Sequential Minimal Optimization (SMO) is a simple algorithm that can quickly solve the above problem. SMO decomposes the overall optimization problem into sub-problems. SMO chooses to solve the smallest possible optimization problem at every step. For this problem, the smallest possible optimization problem involves two Lagrange multipliers, because the Lagrange multipliers must obey a linear equality constraint. At every step, SMO chooses two Lagrange multipliers to jointly optimize, finds the optimal values for these multipliers, and updates the SVM to reflect the new optimal values.

5.2.1 Solving for two Lagrange multipliers:

From the constraints, we know that α_1 and α_2 must lie within the box [0, C] x [0, C] shown. Depending on what the line is there will be generally a lower-bound L and higher bound H on allowed value of α_2 to ensure that α_1 and α_2 lies in the range [0, C] x [0, C].

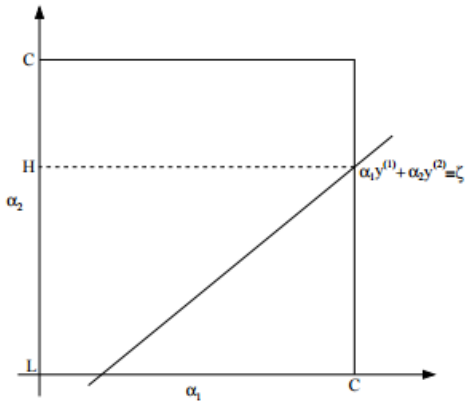


Figure 4. A plot illustrating bound on α_1 and α_2

In order to solve for the two Lagrange multipliers, SMO first computes the constraints on these multipliers and then solves for the constrained minimum. The algorithm first computes the second Lagrange multiplier α_2 and computes the ends of the diagonal line segment in terms of α_2. If the target y_1 does not equal the target y_2, then the following bounds apply to α_2:

$$L = \max(0, \alpha_2 - \alpha_1), \qquad H = \min(C, C + \alpha_2 - \alpha_1). \tag{4}$$

If the target y_1 equal the target y_2, then the following bounds apply to α_2:

$$L = \max(0, \alpha_2 + \alpha_1 - C), \qquad H = \min(C, \alpha_2 + \alpha_1). \tag{5}$$

The second derivative of the objective function along the diagonal line can be expressed as:

$$\eta = K(\vec{x}_1, \vec{x}_1) + K(\vec{x}_2, \vec{x}_2) - 2K(\vec{x}_1, \vec{x}_2). \tag{6}$$

Under normal circumstances, the objective function will be positive definite. In this case, SMO computes the minimum along the direction of the constraint:

$$\alpha_2^{\text{new}} = \alpha_2 + \frac{y_2(E_1 - E_2)}{\eta}, \tag{7}$$

where $E_i = wx_i + b - y_i$ is the error on the ith training example. As a next step, the constrained minimum is found by clipping the unconstrained minimum to the ends of the line segment:

$$\alpha_2^{\text{new,clipped}} = \begin{cases} H & \text{if} \quad \alpha_2^{\text{new}} \geq H; \\ \alpha_2^{\text{new}} & \text{if} \quad L < \alpha_2^{\text{new}} < H; \\ \text{L} & \text{if} \quad \alpha_2^{\text{new}} \leq L. \end{cases} \tag{8}$$

Now, let $s = y_1 y_2$. The value of α_1 is computed from the new, clipped α_2:

$$\alpha_1^{\text{new}} = \alpha_1 + s(\alpha_2 - \alpha_2^{\text{new,clipped}}). \tag{9}$$

SMO will move the Lagrange multipliers to the end point that has the lowest value of the objective function.

5.2.2 Checking the convergence:

In order to check convergence of the algorithm, KKT conditions are checked to be within ε of fulfillment. Typically, ε is set to be 0.01. We have also set the ε value to be the same. The SMO algorithm is set to converge quickly under this value.

5.2.3 An Optimization for SVMs:

To compute a linear SVM, only a single weight vector w needs to be stored, rather than all of the training examples that correspond to non-zero Lagrange multipliers. If the joint optimization succeeds, the stored weight vector needs to be updated to reflect the new Lagrange multiplier values. The weight vector update is:

$$\vec{w}^{\text{new}} = \vec{w} + y_1(\alpha_1^{\text{new}} - \alpha_1)\vec{x}_1 + y_2(\alpha_2^{\text{new,clipped}} - \alpha_2)\vec{x}_2. \quad (10)$$

Algorithm:

Input: C: Regularization parameter
ε: numerical tolerance
max_iter: maximum number of iterations
$(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$: Training data

Output: $a_1, a_2, \ldots, a_m$: Lagrange multipliers for solution
b: Threshold for solution
w: normal vector to hyperplane

- Initialise $\alpha_i = 0 \; \forall i, b = 0$
- Initialise iterations = 0
- **while**(iterations < max_iter):
  - Iterations = iterations + 1
  - **for** i = 1,…,m
    - Select $j \neq i$ randomly
    - Select a_i, a_j
    - Select x_i, y_i, x_j, y_j
    - Compute L and H using (4) and (5)
    - Calculate η using (6)
    - Calculate w and b using constraints (3)
    - Compute error E_i, E_j
    - Set new value of α_j using (7) and (8)
    - Determine new value α_i using (9)
    - Check for convergence using ε
  - **end for**
- **end while**
- return b and w

To predict:

- return $sign(w^T x + b)$ for every x ϵ testing data

5.3. Naïve Bayes:

Naïve Bayes is based on applying Bayes' theorem with the 'naïve' assumption of independence between every pair of features. Bayes' theorem states the following relationship

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots x_n \mid y)}{P(x_1, \ldots, x_n)} \tag{11}$$

Since $P(x_1, \ldots, x_n)$ is constant given the input, we can use following classification rule:

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y), \tag{12}$$

For continuous values, the likelihood of features is assumed to Gaussian:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) \tag{13}$$

5.4 Logistic Regression:

Logistic Regression is a type of regression that predicts the probability of occurrence of an event by fitting data to a logit function (logistic function). The logistic regression hypothesis is defined as:

$$h_\theta(x) = \frac{1}{1+e^{\theta^T x}} \tag{14}$$

The cost function and gradient for logistic regression is given as below:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}[-y^i \log(h_\theta(x^i)) - (1 - y^i)\log(1 - h_\theta(x^i))] \tag{15}$$

5.5 K-Means Algorithm:

K-means classifiers are based on learning by comparison of given test data with training data. For a testing data, a K-means classifier seeks to detect a group of k-objects in the training set that are closest to the unknown data and the label of the unknown data is based on predominance of a specific class in the neighborhood.

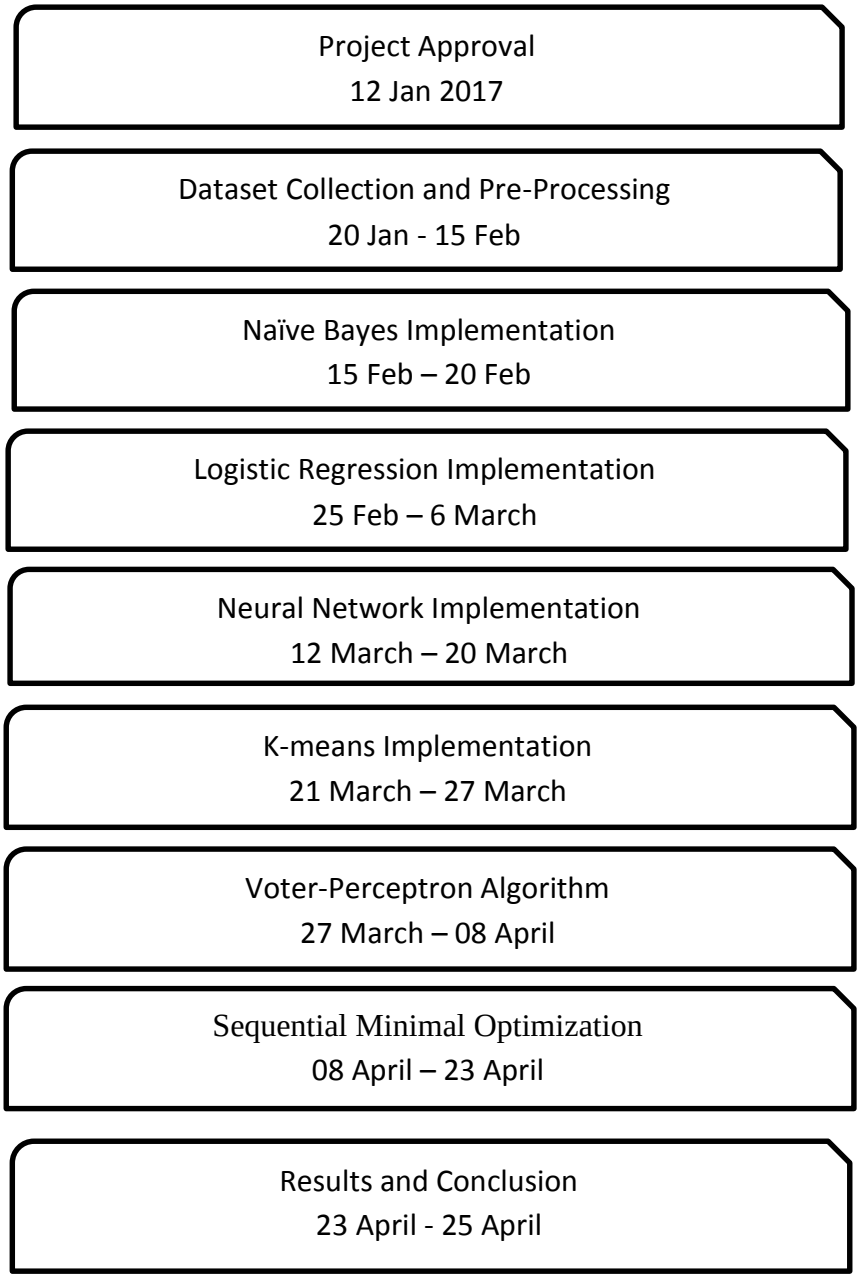
6. Hardware and Software Requirements

6.1. Programming Language: Python

6.2. Pandas Library (Python): Python library for data manipulation

6.3. Scikit-learn (Python): Python library for machine learning

7. Activity Time Chart



8. Results

In this project, 50% data has been used for training and 50% data for testing purpose. The data has randomly divided into training and testing set. 10 instances of each algorithm were taken into account and averaged results were calculated.

8.1 Voted-Perceptron Algorithm:

The voted-perceptron algorithm is trained on both normalized and unnormalized data to check whether the normalization actually helps in improving results. The value of T is also varied from 1 to 100 to get improved result. The results are depicted in the table 1 and table 2.

Iterations	Support Vectors	Mistakes	Precision	Recall	F1-Score	Accuracy	AUC
1	377	377	0.7474	0.7407	0.7312	0.7407	0.7147
2	705	705	0.7511	0.7497	0.7419	0.7497	0.7232
3	974	974	0.7778	0.7738	0.7687	0.7738	0.7539
4	1322	1322	0.7976	0.7918	0.7864	0.7918	0.7706
5	1589	1589	0.7800	0.7763	0.7711	0.7763	0.7578
10	2821	2821	0.8036	0.8094	0.8005	0.8034	0.7866
15	4203	4203	0.8124	0.8129	0.8115	0.8129	0.7994
20	5342	5342	0.8134	0.8139	0.8134	0.8139	0.8039
25	6133	6133	0.8286	0.8289	0.8288	0.8289	0.8212
50	12862	12862	0.8209	0.8209	0.8203	0.8209	0.8128
75	17842	17842	0.8428	0.8430	0.8418	0.8430	0.8316
100	**24120**	**24120**	**0.8507**	**0.8510**	**0.2808**	**0.8510**	**0.8447**

Table 1. Results of Voted-Perceptron Algorithm on unnormalized training set

Iterations	Support Vectors	Mistakes	Precision	Recall	F1-Score	Accuracy	AUC
1	250	250	0.8606	0.8600	0.8599	0.8600	0.8545
2	436	436	0.8682	0.8681	0.8672	0.8681	0.8589
3	634	634	0.8646	0.8635	0.8635	0.8635	0.8583
4	792	792	0.8805	0.8806	0.8805	0.8806	0.8765
5	966	966	0.8989	0.8987	0.8985	0.8986	0.8941
10	1577	1577	0.9014	0.8996	0.9000	0.8997	0.9005
15	2151	2151	0.9052	0.9052	0.9051	0.9052	0.9021
20	2826	2826	0.9272	0.9267	0.9269	0.9267	0.9254
25	3220	3220	0.9348	0.9347	0.9347	0.9348	0.9323
50	**5259**	**5259**	**0.9409**	**0.9407**	**0.9408**	**0.9408**	**0.9395**
75	6611	6611	0.9405	0.9403	0.9404	0.9403	0.9390
100	7693	7693	0.9325	0.9320	0.9322	0.9323	0.9289

Table 2. Results of Voted-Perceptron Algorithm on normalized training set

After several iterations, the voted-perceptron algorithm tend to converge to a consistent predictor vector. “Support Vectors” tell us the size of all instances on which the mistake has occurred during training. Since on running more iterations it tends to gain experience and also find more mistakes on prediction vector, the value of “support vectors” (mistakes) should increase with number of iterations. This intuition has also been supported by the obtained results.

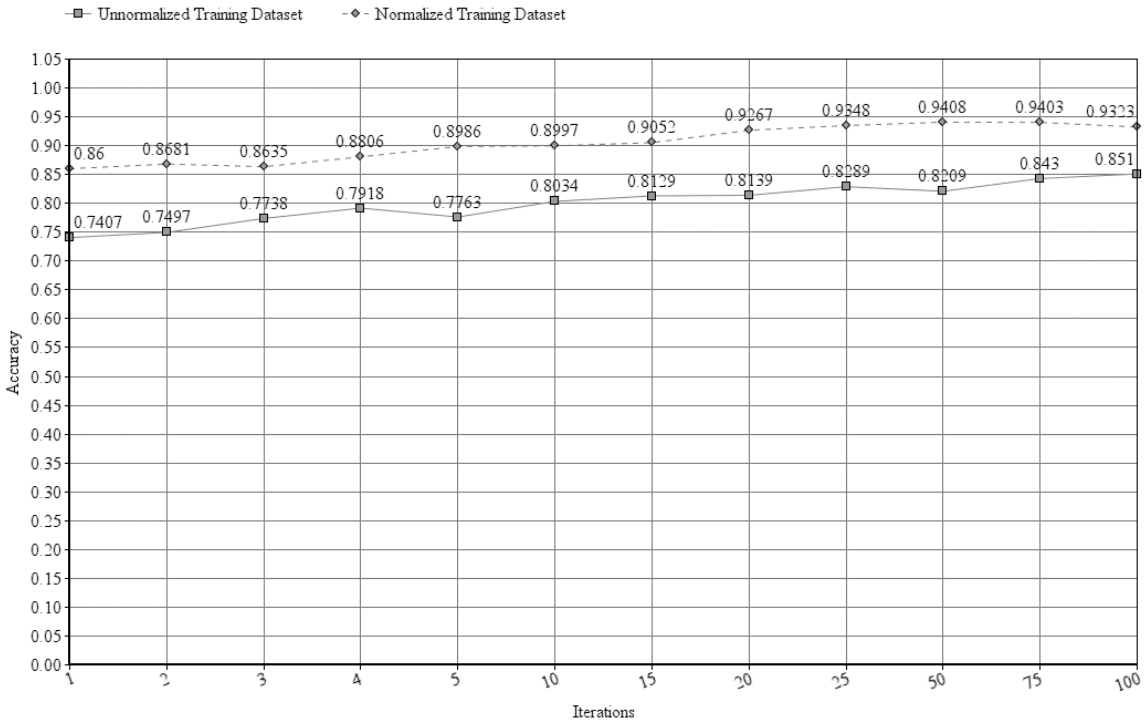


Figure 5. Plot illustrating result of voted-perceptron algorithm

The normalized training set makes less number of “mistakes” in comparison to unnormalized training set for given value of iteration. This shows that normalizing the data actually helped us in improving the results.

It has also been observed that after 50 iterations the algorithm tends to overfit the data and the accuracy decreases with increasing number of iterations past 50. The best accuracy is found to be 94.08% using 50 iterations.

8.2 Sequential Minimal Optimization:

The SVM has given an objective of predicting the *CrimeStatus* of a state. There are total 1994 instances in our dataset. Two different SVM’s are trained for this problem: a linear SVM and a quadratic SVM. Sequential Minimization Algorithm is used to solve the SVM. The results are obtained for different values of epsilon, C and iterations. Table 3 illustrates the results.

Kernel	Epsilon	C	Iterations	Precision	Recall	F1-Score	Accuracy	AUC
Linear	0.01	0.1	1000	0.8530	0.8490	0.8498	0.8490	0.8513
Linear	0.01	0.1	10000	0.8589	0.8570	0.8575	0.8570	0.8562
Linear	0.01	1.0	1000	0.8642	0.8635	0.8637	0.8635	0.8623
Linear	0.01	1.0	10000	0.8695	0.8660	0.8666	0.8660	0.8676
Linear	0.001	0.1	1000	0.8565	0.8545	0.8550	0.8545	0.8543
Linear	0.001	0.1	10000	0.8645	0.8640	0.8642	0.8640	0.8608
Linear	0.001	1.0	1000	0.8572	0.8515	0.8524	0.8515	0.8554
Linear	0.001	1.0	10000	0.8795	0.8786	0.8788	0.8786	0.8766
Quadratic	0.01	0.1	1000	0.8729	0.8731	0.8724	0.8731	0.8641
Quadratic	0.01	0.1	10000	0.8736	0.8736	0.8730	0.8736	0.8656
Quadratic	0.01	1.0	1000	0.9168	0.9162	0.9157	0.9162	0.9088
Quadratic	0.01	1.0	10000	0.9186	0.9167	0.9159	0.9167	0.9068
Quadratic	0.001	0.1	1000	0.8755	0.8756	0.8753	0.8756	0.8702
Quadratic	0.001	0.1	10000	0.8731	0.8731	0.8724	0.8731	0.8626
Quadratic	**0.001**	**1.0**	**1000**	**0.9248**	**0.9232**	**0.9226**	**0.9232**	**0.9149**
Quadratic	0.001	1.0	10000	0.9166	0.9152	0.9145	0.9152	0.9064

Table 3. Results of SMO Algorithm on normalized training set

As depicted in the table, the precision and recall are higher in quadratic kernel than linear kernel. Although the result is not much different, quadratic kernel with epsilon = 0.001 and C = 1 with iterations limited to 1000 gives the best possible result.

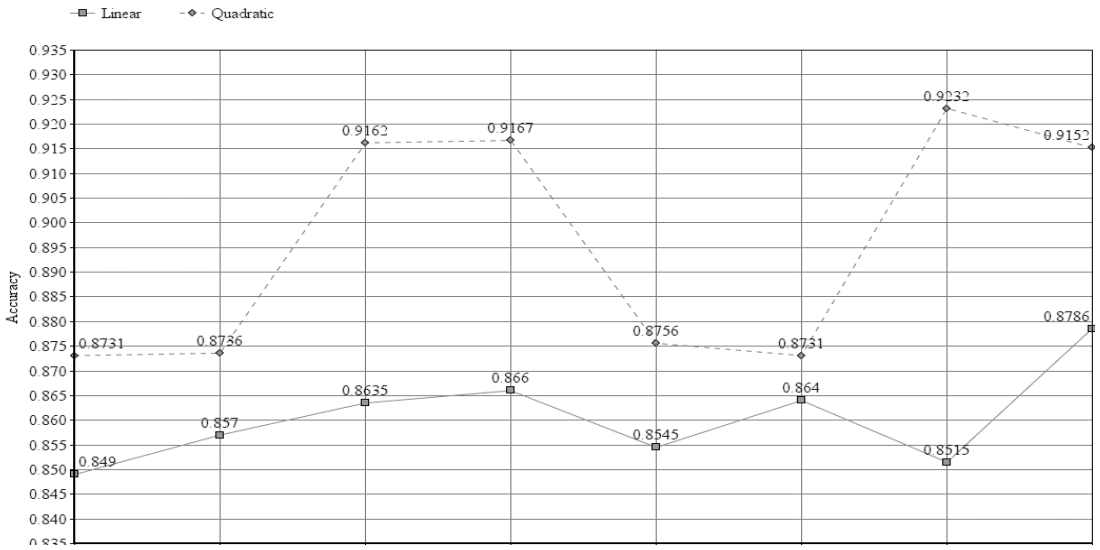


Figure 6.Plot illustrating results of SMO Algorithm on normalized training set

8.3 Comparison:

Evaluation of selected classification algorithms is conducted by comparing the precision, recall, accuracy and f1-score. Precision shows the proportion of data that has been classified correctly. Recall represents the percentage of information which is relevant to class and id correctly classified. Accuracy is the percentage of instances that were classified correctly by classifier. Table 4 illustrates the comparison of different algorithms.

Algorithm	Precision	Recall	F1-Score	Accuracy	AUC
Naive Bayes	0.8575	0.8539	0.8515	0.8539	0.8386
Logistic Regression	0.8955	0.8954	0.8921	0.8954	0.8937
SVM using sklearn	0.8030	0.8012	0.7959	0.7692	0.7955
K-means	0.4567	0.4361	0.4384	0.4361	0.4415
K-means using sklearn	0.4555	0.4377	0.4401	0.4377	0.4413
Voted-Perceptron	**0.9409**	**0.9407**	**0.9408**	**0.9408**	**0.9395**
SMO Algorithm	0.9248	0.9232	0.9226	0.9232	0.9149

Table 4. Comparison of Different Classification Algorithms

As shown in table 3, the voted-perceptron algorithm gives the best classification results followed by SMO Algorithm. The logistic regression (0.89) and Naïve Bayes (0.83) are also better classification algorithm with accuracy over 80%. Standard SVM and K-means however are not preferred for this classification.

Naïve Bayes and Logistic Regression results are differentiated from SMO and Voted-Perceptron Algorithm. For binary classification, AUC is a better measure of performance and a classifier with higher AUC is said to be better. Clearly, AUC tells us that SMO and Voted-Perceptron are better classification algorithms for this problem than Logistic Regression and Naïve Bayes.

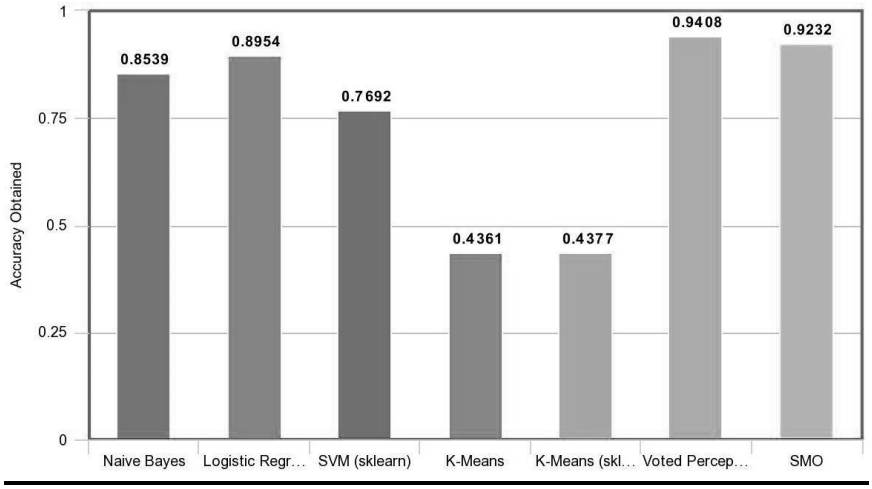


Figure 7. Plot illustrating comparison of classifiers on the basis of accuracy

9. Conclusion and Future Scope

The aim of this project was to perform classification of the given dataset into binary categories, “Critical” and “Non-Critical”. In this project, Voted-Perceptron and Sequential Minimal Optimization algorithms are implemented and the results are compared with different classifiers to determine more accurate classifier. We have shown via results that Voter-perceptron Algorithm presents the best accuracy. From the experimental results, Voted-Perceptron with T = 100 gives the best performance followed closely by SMO Algorithm. Based on the results, Voted-Perceptron and SMO are strong candidates for crime related classifications and better than K-means, Logistic Regression and Naïve Bayes.

Due to the huge size of criminal activities data, the pattern detection and prediction of crime status is almost impossible for human investigators irrespective of their years of experience. By using Classification Algorithms to increase the efficiency as well as precision, police department and investigators can allocate their time to other valuable tasks. Predicting the crime-prone areas can also help in efficient distribution of police forces to reduce crime rate of a state.

10. References

[1] Anna L. Buczak, Christopher M. Gifford, “Fuzzy Association Rule Mining for Community Crime Pattern Discovery”, In ACM SIGKDD Workshop on Intelligence and Security Informatics (ISIKDD '10), 2012.

[2] Chung-Hsien Yu, Max W. Ward, Melissa Morabito, Wei Ding, “Crime Forecasting Using Data Mining Techniques”, In Proceedings of the 2011 IEEE 11th International Conference on Data Mining Workshops (ICDMW '11), 2011

[3] Donald E. Brown, “The Regional Crime Analysis Program (RECAP): A Framework for Mining Data to Catch Criminals”, In Proceedings of the International Conference on Systems, Man, and Cybernetics, 1998.

[4] Shyam Varan Nath, “Crime Pattern Detection Using Data Mining”, In Proceedings of the International Conference on Web Intelligence and Intelligent Agent Technology, 2006.

[5] Hsinchun Chen, Wingyan Chung, Jennifer Jie Xu, Gang Wang, Yi Qin, Michael Chau, “Crime Data Mining: A General Framework and Some Examples”, Computer, IEEE Computer Society Press, 2004.

[6] John C. Platt, “A Fast Algorithm for Training Support Vector Machines”, Appeared in Kernel Methods of Support Vector Learning, MIT Press, 1998.

[7] Freund, Y. and Schapire R. E. “Large Margin Classification Using Perceptron Algorithm”, Machine Learning, 1998.

[8] Rosenblatt, “The Perceptron: A Probabilistic Model for information storage and organization on brain”, Psychological Review, MIT Press, 1998.

[9] Christopher J.C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," Appeared in Data Mining and Knowledge Discovery 2, 121-167, 1998

[10] UCI Machine Learning Repository, http://archive.ics.uci.edu/ml/datasets/Communities+and+Crime+Unnormalized#

11. Remarks