SPM4450: Fundamentals of Data Analytics
Assignment 3

Structuring data analysis methods for developing efficient governance in dynamic socio political systems

Case Study of Indian Caste Reservation System

Authors:
Rhythima Shinde (4516389), Bramka Arga Jafino (4516516), Paraskevi Kokosia (4417062)

Course instructor and guidance:
Prof. Jan van der Berg

Abstract

Today's socio-political systems have the property of dynamism due to the ever-increasing amount of information generated and transferred by interactions among stakeholders. With the development of data science field of study, gathering and studying this information have not just enabled us to access more knowledge but also given us the power to predict our future based on the evolving patterns in the history. Thus, this paper discusses different types of predictive models in one-class classification techniques, namely: Support Vector Machine, Naïve Bayes, Principal Component Analysis, and kNN algorithms, for analyzing the reservation system in India. The research aims to check the efficiency of the reservation system, especially between the northern and southern states of India, by checking if the socioeconomic attributes of the castes and tribes determine whether the caste/tribe is reserved or not. The results depict the dependence of the accuracy of the model on not only the algorithm but also the demographics and attributes of the population. Thus, this research tries to understand with the help of a pressing issue about how the governance of such dynamic societal systems can be made more efficient using the technological tool of data science.

Keywords*: Dynamic societal systems, One-class classification, Reservation system, Demographics/Attributes, Support Vector Machine, Principal Component Analysis, Naïve Bayes, k-Nearest Neighbors*

Contents

1. Introduction

During the times India was ruled by the British, they used the historical social classes of the people to smoothen administrative procedures (only the one with a higher social class were given a job). This hierarchy of classes was defined, based on the occupation the people had. In 1947, when India became independent, India committed to develop policies that will improve the wellbeing of the people oppressed by hierarchical caste system. The disadvantaged castes who historically have suffered from such discrimination due to the hierarchical caste system got facilitated using system of "reservation" from 1950s [1]. For example, in education, reservation means using a quota, where school vacancies must be reserved for members from these castes. This reservation system has been allocated based on the demographics of these castes (i.e. income, education level, etc.) and is used to decide which castes need support and thus would be labeled as "scheduled" or "reserved". Until today no insight has been gained on the efficiency and the accuracy of the reservation caste system in India. There have been many debates over the existence of this reservation system today as the reservation system is no more allocated on the basis of demographics but only based on the historical record of the parents' castes [2]. To complicate matter, the socioeconomic conditions of these castes have changed vastly now too.

These debates pose some questions regarding the reservation system efficiency itself. Thus, this project aims at answering the research question that "Is the reservation system for scheduled caste and scheduled tribe equally efficient for different states and is it well defined by the socio-economic indicators?" To answer this we dive into following research questions:

1) For the scheduled castes and tribes, does there exist a difference in the northern states and the southern states for efficiency of reservation system?
2) Are the socioeconomic conditions in the all states justifying the reservation system in India?
3) What can be an effective way to define reservation system in India for future based on the present data?

The objective of this research which is based on the demographics in 2011, given by the Census of India, is to obtain an understanding of the caste reservation system and its efficiency. This will permit us to apply a prediction model for the reserved caste and tribes based on the India's demographics. Along with that, a series of different algorithms will be tested out in order to find out how accurate a model is and if it could be used for future predictions and in other socio political systems with similar structure.

The paper started by giving a historical overview of the reservation system and its problem description. Chapter 2 lays down the conceptualization and methodology followed for answering the posed research questions. Chapter 3 includes the data analysis part. We begin with preprocessing our dataset and continue by comparing and carefully selecting the algorithms that will be tested out. In the validation part one class Support Vector Machine (SVM), one class Naïve Bayes (ONB), K-nearest neighbor (kNN) and Principal Component Analysis (PCA) are selected and tested out respectively. Then, the results obtained from the methods are presented and analyzed in Chapter 4. Finally, brief conclusions and reflection of the main result obtained and discussion for future work that can be undertaken, follow in Chapter 5.

2. Conceptualization and Methodology

In order to answer the questions formulated before, we take following steps in data analytics to understand the models' accuracy in dynamic socio-political systems:

1) The first question can be answered by understanding the efficiency of the caste reservation system by training the dataset by using the data from the northern states on the dependent variable of the "caste allotted" and the independent variables of demographics like "income", "education level", "health conditions" etc. The trained model will then be tested on the data from the southern states.
2) The second question can be answered by the same algorithm like the first one. The difference is that the training dataset and the test dataset come from randomly splitting the whole data (combining both northern and southern state) into two separate datasets.
3) If the model has high accuracy, it cannot be concluded that the reservation system is efficient enough, because there are inaccuracies in the model itself. Thus, we chose more models to carry out this predictive analysis. If all the models have a high accuracy, it can be considered that the reservation system is efficient enough. If some models peculiarly fail, those models will be investigated in comparison to those who have a good accuracy. If all models fail, then it can be said that there is some inherent flaw in the reservation system itself and the debates around the existence of reservation systems are legit enough. In this last case, the most accurate predictive model will be suggested for finding an efficient reservation system.

This study helps finding models with good prediction performance which can work for such inefficient socio-political systems. This will involve comparison of several algorithms with possibly different sub sample sets of the available data. The dataset used here has the following attributes: age, gender, education, income, and it will belong to all states, but divided into northern and southern state. The method for the decisions is followed as shown in the Figure 1 below.

The predictive modelling consists of three different steps and each step includes a different dataset: the initial, the validation and the test. In the first step the initial dataset begins with the definition of the problem that has to be solved. Then follows the collection of data related to the problem and evaluation of the existing models or algorithms is executed. In the second step the model has to be selected carefully based on the objectives that have been defined and the data that will be used. Then the model is tested out. The third step involves the application of the model where the prediction is made.

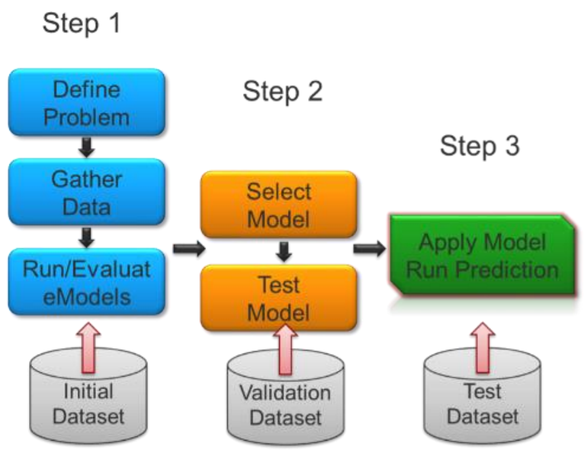


Figure 1: Steps to carry out for predictive modelling [3]

3. Data analysis

3.1. Pre Processing of data

Available indexes were given in Census of India [1] for the year 2011 and this is used as the only data source. The description of the indicators of the dataset is provided in detail in various government run platforms which includes demographics, and other attributes. All states' data is used here to get a clearer idea. A later distribution is done on the north and south states of India to see how the two regions are different in their effective use of reservation system. Datasets used here are based on employment, education, social and healthcare indicator and are extracted from the available census data by creating excel macros, and the process is explained in detail in Appendix 1. Also, some castes' data was not available for some attributes, so such castes were cleaned completely as it would lead to bias, e.g. the castes with 0 number of surviving children were removed. The main indicators are shown below in Table 1.

Table 1 Demographics or the attributes used for the analysis

Index	Description
IPTP	Illiterate percentage in the caste
PELP	Students in primary education per literate in the caste
GALP	Graduate and above educated per literate in the caste
SMLP	Secondary or matric educated per literate in the caste
SMSC	Total surviving male per total surviving children
SCMW	Total surviving children per total married women
CLTC	Percentage of child-labor in the caste
TWTP	Total employment rate in the caste
LOC	1 = North, 2 = South

The indicators (attributes) are chosen such that it can be seen that how backward a caste is, and based on the assumption that a backward caste should only technically demand reservation in an efficient reservation system. The educational indicators comprise of illiterate percentage, students in primary education per literate, graduate and above educated per literate, and secondary/ matric educated per literate in the caste to understand the educational levels distribution in the caste. The distribution is important to be understood because the level of education describes the level of progress or backwardness of caste (with the assumption that more "backward" caste is one with lower education, economic and health level). The health indicators comprise of studying total surviving children per total married women to understand the average health of a woman and a child. This also helps in understanding the willingness to carry more than two children, which is associated to traditional beliefs against contraception, but this is not the target of study here. Total surviving male per total surviving children is a social indicator which helps in understanding if a wide discrepancy in the female and male population (relating to female genocides/ feticide) relates to the backwardness of a caste and thus demands more reservation. The employment indicators include the percentage of child-labor in the caste (forcing children to work due to the poverty or lack of education in the families) and the total employment rate.

3.2. Choice of Models/Algorithms

Predictive analysis models use various statistical, data mining and machine learning algorithms. The first step for a model to succeed is to define the objectives of the model and select the data that will be used. A model can represent a query, a decision tree or an advanced mathematical analysis. A number of different algorithms should be tested out in order to find out which variables in the data have predictive power .There are three basic forms of predictive algorithms: classification, clustering and regression [3, 4].

Classification is a supervised learning method which predicts categorical class labels and classifies the data based on the training set and the values in a classifying attribute named as class and uses it to classify and the new data to the respective class [5]. Clustering creates different groups (clusters) which consist of a set of objects that are quite similar to each other or appear similar characteristics in contrast with other clusters. Regression is a supervised learning method which is mainly used for continuous response values.

In order to decide whether a caste fits in the reservation system or not we have decided to apply classification algorithms for our case, as here the main question is to compare between the reserved/scheduled (positive class) and non-reserved/non-scheduled class (negative class). Due to limitation in our dataset (availability of only data related to the scheduled castes and tribes and no data available for the non-scheduled castes), one class classification will be used. One class classification tries to find out the objects of a specific class amongst all objects, by learning from a training set which contains the objects of that class. In contrast with two class classification here only information of target class is available. Here, a classifier is learned from only positive and unlabeled samples in a semi supervised way.

One class classification can be applied in three different ways, using density estimations, boundary methods and reconstruction methods. The goal in density estimation is to estimate the density of the vector that has to be observed e.g. in Gaussian model and Parzen density estimators. Boundary methods find the distance from the vector to a learnt boundary and is based on numeric data and can be applied through Support Vector Data Description (SVDD) and K-centers. Reconstruction method calculates the reconstruction error and the algorithms vary based on the category of data [6]. In Appendix 2, the advantages of each one class classifiers method can be found in detail.

Based on carious literature studies i.e. the comparison between the advantages and the suitability of each algorithm between SVM or Support Vector Data Description(SVDD) [7], One class naïve Bayes [8, 9, 10], Expectation Maximization Algorithm(EM) [11], Global Gaussian approximation and Parzen density estimation [12], 1-Nearest Neighbor method [12, 13], Decision tree [14, 15, 16], Neural networks, K-means [7], Gaussian data description [7], Principal Component Analysis(PCA) [7] led to the selection of the final methods choices as: Support Vector machine and k-nearest neighbor(boundary method), Naïve Bayes(density method), Principal Component Analysis (reconstruction method).

The selection of classifiers was based on two aspects: First, the type of the features which the classifier takes as input and the characteristics of the classification method. A set of parameters such as the number of training examples, the number and independency of features, linearity, space dimensionality, training time, sensitivity to the errors in training data, accuracy had also first to be checked out in our dataset in order to select these specific algorithms [17, 18, 19, 20]. In terms of accuracy, in no free lunch theorem, it is stated that no single classification method can offer superior accuracy without depending on the context that it is applied [21].

3.3. Applications of models

3.3.1. Method 1: Support Vector Machine

Support Vector Machine (SVM) basically tries to build hyperplane that best separates the high-dimensional input vectors for classification purpose [22]. It is therefore considered as supervised learning technique as it requires training dataset to map the unknown and nonlinear dependency between the input vectors and their scalar output [23]. Nevertheless, current application of SVM has also evolved to tackle one class classification purpose. The constructed hyperplane acts as a decision boundary that separates the positive data and the non-positives [24].

Further researches have also been conducted on how different kernels, i.e. linear, polynomial, radio based function, and sigmoid, can be applied to the algorithm.

While the traditional SVM utilizes penalty parameter C to determine the extent of the soft margin [23], One-class SVM (OSVM) uses parameter *v* as both an upper limit of the outliers' fraction and the lower limit of the support vector fraction [25]. Therefore, the parameter *v* ranges between zero and one. While cross-validation and grid search can be used to optimize this parameter, both techniques require the presence of negative data in the training set. As no negative data presents in this case study, sensitivity analysis is used instead to derive insight for answering the research question.

In order to implement the OSVM, built-in module OneClasSVM from class sklearn.svm in python is utilized. Firstly, the dataset from states which belong to 'North' category is used as the training set. The model is then applied to the dataset from states which belong to 'South' category by initially setting the *v* parameter to 0.1. This implies that it is assumed that the maximum fraction of training errors allowed is 10%. Put it differently, it is assumed that at maximum 10% of the data points in the training set is considered as non-positives. The percentage error of the test dataset (the 'South' states) are then compared to the percentage error of the training dataset. As it is difficult to derive conclusion from one observation, sensitivity analysis is then conducted by running the model 20 times, taking parameter *v* between 0.01 and 0.2. In this way, the discrepancy between the percentage error of the two datasets can be better investigated to derive conclusion.

The second analysis comes from conducting split test: randomly dividing the whole dataset regardless of their 'North' and 'South' category. The split test uses 80:20 rule, 80% of the data points in the dataset is used as the training set while the remaining 20% acts as the test set. The aim of this split test is twofold: (i) studying the discrepancies of the attributes of the scheduled caste/tribe in random order instead of seeing the North-South states difference, and (ii) evaluating the sensitivity of the algorithm. The split test is conducted ten times, resulting in ten different combinations of training and test datasets. The same sensitivity analysis is done for each of the replication.

3.3.2. Method 2: One class Naïve Bayes

Naïve Bayes algorithm is a classification method which makes prediction based on the probabilities of each attributes in the dataset fitting into each class.There are two models used: the multi-variate Bernoulli model and the multinomial model.
In the multi-variate Bernoulli model, a vector of binary attributes is used to represent a document, indicating whether the command occurs or doesn't occur in the document.
The multinomial model uses the frequency of each command that occurs to represent a document, which is called "bag of words" approach capturing the word frequency information in documents.

Classification by using Naïve Bayes generally consists of three steps: (i) calculating the Gaussian probability density function to get the conditional probability of an attribute value belonging to a given class value, (ii) calculating the probabilities of all of the attributes at once for a data entry and returns the probability of the data entry belonging to the class, and (iii) observe the highest probability and return the accompanying class. Based on the third step, the Naïve Bayes algorithm is therefore normally used for two-or-more classes classification.

Though, recent research has shown application of Naïve Bayes classifier for anomaly detection or one-class classification [26]. The One-Class Naïve Bayes (ONB) also works in three steps: (i) determining the Gaussian probability density function of the training dataset (which is basically step (i) and (ii) in the previous paragraph), (ii) calculating the target rejection threshold T, and (iii) determine whether each data instance in the test dataset has higher or lower probability score compared to threshold T. If the test data instance has higher probability score, it is considered as positive data, and vice versa. As for the threshold T, Li, et al (2010) used the minimum probability score from the training dataset as the cutoff point.

The ONB analysis in this case study was conducted in two ways. Firstly, the similar sensitivity analysis logic was executed by using different threshold T values toward 'North' and 'South' states. As there is ongoing debate with regard to the appropriateness of the scheduled caste and tribe system, the assumption made in the analysis is that there is certain percentage of the currently scheduled castes and tribes, which if based on their attributes, are overqualified to be categorized as scheduled. The implication to the ONB is instead of using minimum probability score from the training dataset as the cutoff T threshold, a range between 1st percentiles to 20th percentile with an interval of one is employed for determining the threshold value. The percentage of the train dataset which is classified as positive class is then recorded and compared to the threshold percentage. The second analysis also stems from the same argument as the second analysis of OSVM. Ten random data splits were executed to tear the whole dataset into training and test dataset.

The ONB presents a few limitations regarding the probability estimation along with the assumption that there is at least one negative class to estimate the probability given the negative class. In cases that the variables of the dataset are continuous the algorithm might underperform. Moreover, in high dimensions curse of dimensionality is appeared for the assumption of the algorithm. In general for one class classification when it exists the assumption for the existence of the negative class, the algorithm is ineffective [27].

3.3.3. Method 3: K nearest neighbor

The k-nearest neighbour is a simple non parametric classifier which selects the k samples whose variables values are nearer than the sample that has to be classified and use them to estimate the most common class of the sample through the majority voting procedure of its neighbours. If K = 1, then the case is simply assigned to the class of its nearest neighbor. The one class nearest neighbour classifier (KNN) is an alternative modification of k nearest neighbour as it learns from positive examples only. It operates by storing all the training examples as its model and it calculates the distance to its nearest neighbour, for a given test example. The k nearest neighbour uses the principle of the local density estimation which is expressed by the formula as shown in the equation 1 below, where k is the number of nearest neighbors accounted, N is the total number of points, V is the volume of the 8 dimensional hypersphere (as there are 8 attributes) with radius and the distance between the NN(z), i.e. is the nearest neigbour and the point z.

$$p_{\mathrm{NN}}(\mathbf{z}) = \frac{k/N}{V_k(\|\mathbf{z} - \mathrm{NN}_k^{tr}(\mathbf{z})\|)}$$

Equation 1: local density of any point z in the k dimensional space

If the local density of a point from testing set is larger than the local density of its nearest neighbor then it is accepted as the class entity (in one class classification), else it is rejected from the class. For one nearest neighbor, the condition looks like as shown in the equation 2.

$$\frac{1/N}{V(\|\mathbf{z} - \mathrm{NN}^{tr}(\mathbf{z})\|)} \geq \frac{1/N}{V(\|\mathrm{NN}^{tr}(\mathbf{z}) - \mathrm{NN}^{tr}(\mathrm{NN}^{tr}(\mathbf{z}))\|)},$$

Equation 2: condition for accepting a point in the class set

In this case, the data is first divided into north and south states and the south states are treated as the testing set, while north as the training set, to check whether the reservation systems are equally (not) efficient in the north and south states. Second, for different values of k, this analysis is repeated. Finally, different split tests are performed on this to check the variation of the results based on the data. For this, a python code is developed which first calculates the nearest neighbors and the local density for the training sets. Then for every value in testing set, the nearest neighbor from training set is calculated using sorting algorithms. Here, the attributes' values help in deciding the distance which is calculated as the modulus of the 8 dimensional vector (8 attributes).

3.3.4. Method 4: Principal Component Analysis

Principal Component Analysis (PCA) is a classical statistical method known as an orthogonal linear transformation, and is applied in data analysis mostly for high dimensions. Its main purpose is to reduce the dimensionality of the data in direction of eigenvectors. This involves calculation of a covariance matrix of a dataset, finding the variances and coefficients, by determining the eigenvectors and the eigenvalues of the matrix. The goal is to minimize the redundancy and maximize the variance. Thus the covariance matrix is used to measure how much the dimensions vary from the mean with respect to each other. It centres the data by removing the mean of each sample vector. Each variable is represented as a vector which indicates how each variable contributes to the principal components. PCA starts with the covariance matrix, where the eigenvectors and eigenvalues are calculated and the eigenvalues are sorted in a descending order. Next the principal components are calculated by multiplying each row of the eigenvectors with the ordered eigenvalues. The eigenvector which has the highest eigenvalue is the main principal component of the dataset.

For the one class classification, PCA can be used by setting a threshold value on the basis of eigenvectors which are set as the mean values. This threshold decides whether a point from testing set should be considered from the same class. Here, there is no training and testing set splitting required for the first run, but rather the dimensions are reduced to 2 dimensional space and then those which are above the set threshold are considered as "anomalies" or outliers, which doesn't belong to that class. Then, cross validation is done to check the model with best accuracy, which involves choosing different threshold values. Here, PCA was implemented in Matlab, first for the north and south regions (where south is the testing set) and then for an overall overview of the method in the dataset (where random thresholds from 0.5 times the standard deviation to 4 times the standard deviation are taken) to get the maximum accuracy model.

4. Results and Interpretation

4.1. One class SVM results

By assuming a *v* value of 0.1, the percentage error of the OSVM for both scheduled caste and scheduled tribe can be seen in 2. The percentage error of the test set for both categories are more than 16% higher compared to the percentage error of the training set. The result implies that there is a significant difference between the attributes values of the 'North' states and the 'South' states. Practically speaking, based on the current situation, the socioeconomic standards used to put a person into scheduled caste/tribe are not relevant as there are 16% of the scheduled caste and tribe from the 'South' states which socioeconomic conditions are different from the 'North' states.

Table 2 Percentage Error Result of OSVM, v = 0.1

	Scheduled Caste	Scheduled Tribe
Percentage error of training set ('North' states)	9.83%	9.69%
Percentage error of test set ('South' states)	25.91%	27.52%

To further affirm this result, sensitivity analysis was conducted by changing the *v* parameter from 0.01 to 0.2. The result shown in Figure 2 tells that for larger value of *v,* the gap between test and training set's percentage errors is getting wider. The percentage error of scheduled tribe's test dataset is twofold of its training set's percentage error for *v* value of 0.2. The gap is even wider for scheduled caste where the test dataset's percentage error is 2.5 times larger compared to its counterpart. This result confirms the analysis from the previous paragraph that the standards for scheduled caste and tribe are not fully relevant based on the distinction of North-South states.

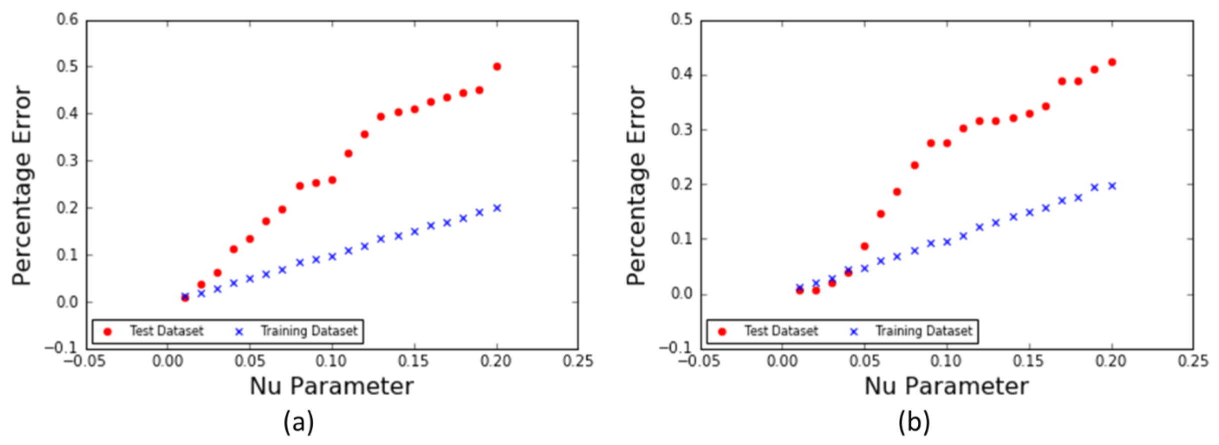


Figure 2: Comparison of Percentage Error Between Test and Training Dataset for (a) Scheduled Caste and (b) Scheduled Tribe

The final analysis comes from doing ten random data splits for the whole dataset and the result can be seen in Figure 3. Though the characteristics of scheduled caste and tribe between 'North' and 'South' states are different, the graph shows that is not the case if all data points are mixed at once. For instance, the percentage error of the test dataset for scheduled caste with *v* parameter of 0.2 ranges between 15% to 25%. Therefore, it can be deduced that the socioeconomic standards for scheduled caste and tribe are still relevant, as the characteristics of the whole group are similar. However, it might also turn out that the results tell that OSVM is an adequate technique to analyze this case. This is because the range of the result, for instance for *v* value equal to 0.2, might indicate the high accuracy of the model.

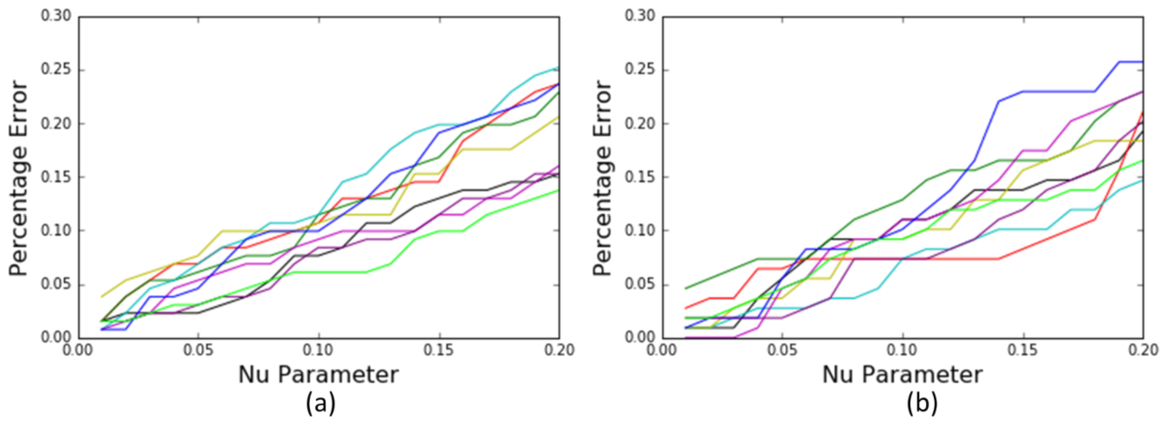


Figure 3 Comparison of Test Dataset's Percentage Error with 10 Random Data Splits for (a) Scheduled Caste and (b) Scheduled Tribe

4.2 One-class Naïve Bayes Result

Using the theoretical threshold T value for the analysis results in a very high percentage of positive-class data for the test dataset (the 'South' states). The scheduled caste's test dataset has 99.5% of positive-class data instances while the rate is 100% for the scheduled tribe. There are two explanations to this result. Firstly, the values of each attributes are normalized values, which for most of the attributes range between 0 and 1. Therefore, creating Gaussian probability distribution to this data will result in a limited range of probability distribution, which tends to create overlap between the training dataset and the test dataset. Secondly, as there might exist some outliers in the training dataset, setting the threshold T value to the minimum value of the training set's probabilities will result in an extremely small value of probability score as it takes the probability score of the outlier. Therefore, there are only small amount of data instances of the test dataset which probability score is lower than the threshold T.

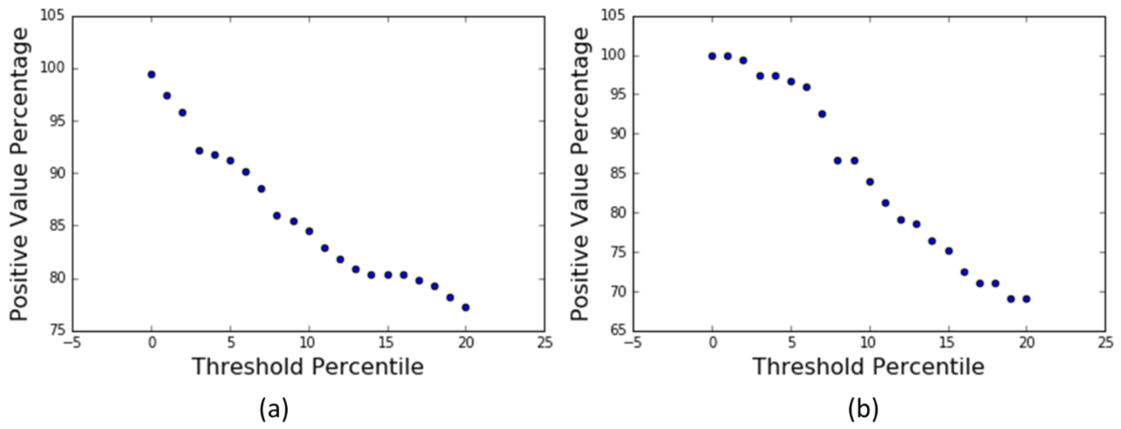


Figure 4: Positive-Class Percentage of the Test Dataset (the 'South' states) for (a) Scheduled Caste and (b) Scheduled Tribe

Based on the two deficiencies explained above, using the theoretical threshold T value of minimum training dataset probability scores is not sufficient for this case study. Therefore, sensitivity analysis is conducted and the result is displayed in Figure 4. The percentage of positive-class data instances from the train dataset decreases along with the increase of the threshold T value as expected. The decrease is quite linear for scheduled caste; as the positive-class percentage does not drop more than 25% when 20th percentile of the training dataset's probability score is utilized. The result is quite different for the scheduled tribe. By assuming that 20% of the training dataset is outliers, that is setting the threshold T value to 20th percentile, the positive-class percentage of the test dataset decreases by around 30%. The 10% difference between the threshold and the resulting positive-class percentage indicates that there is a small magnitude of difference between the attributes of 'North' states scheduled tribes and 'South' states scheduled tribes.

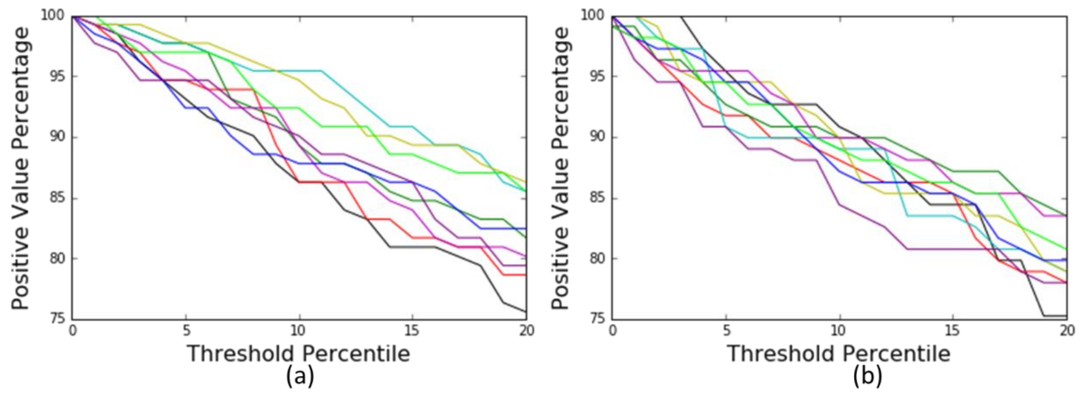


Figure 5 Comparison of Test Dataset's Positive-Class Percentage with 10 Random Data Splits for (a) Scheduled Caste and (b) Scheduled Tribe

Similar to OSVM, ten random data split tests were also conducted for the ONB. The result of the data split tests can be observed in Figure 5. The ranges of both scheduled caste and tribe are quite diverse; for 20th percentile the reduction of the positive-class percentage is between 15-25%. The same explanations like the OSVM data split tests result hold for this outcome, which is either there is a high algorithm accuracy or that there is no significant difference of socioeconomic attributes for the whole scheduled population.

4.3. K nearest neighbor results

The k nearest neighbor results show an increasing accuracy for the increasing value of k as shown in figure 6 below; this is expected as the number of k increases the local density of the training data becomes smaller and thus more testing data fits. With more increasing value of k, the phenomenon of overfitting may arise, but this is not considered in this paper. The training sets here are the northern states and the testing sets are the southern states. Graphs below helps in showing that the number of positive class castes, accepted as the reserved caste are higher for k>3, showing that the reservation system is equally (in)efficient in both of the north and south states.

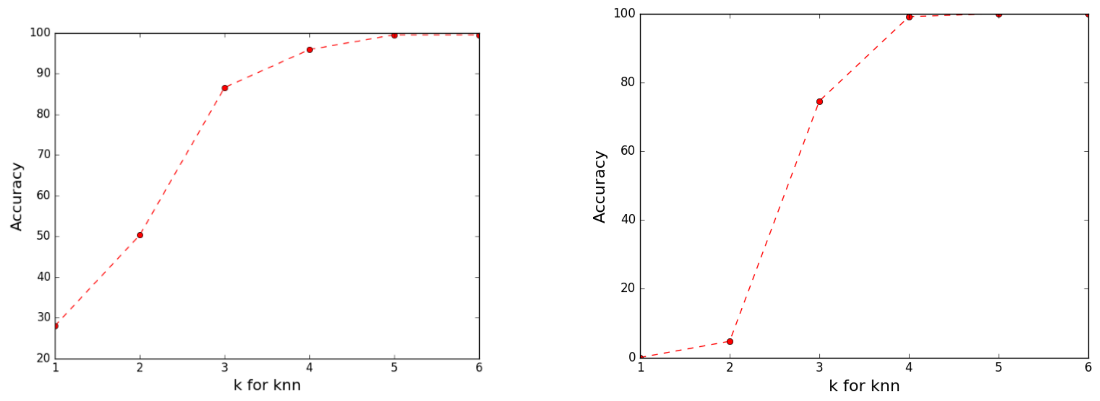


Figure 6: Accuracy versus the value of k in k nearest neighbor (for north versus south states) for (a) Scheduled Caste and (b) Scheduled Tribe

To verify if this holds for the complete dataset, different 4 split ratios with train data of 10%, 20%, 30%, 40%, are taken respectively (randomly data is split in this ratio in train and test are chosen) and the results are plotted in the figure 7 below. The figure shows that the results are very close for different splits, showing that there is either a very high algorithm accuracy or that there is no significant difference of socioeconomic attributes for the whole scheduled population. Usually a method of k fold cross validation is also used here, to verify which value of k is optimum, but here from figure 6, value of k>=3 is chosen, as the accuracy increases after k=2 very highly and it is suitable to be considered and compare with other methods efficiently.

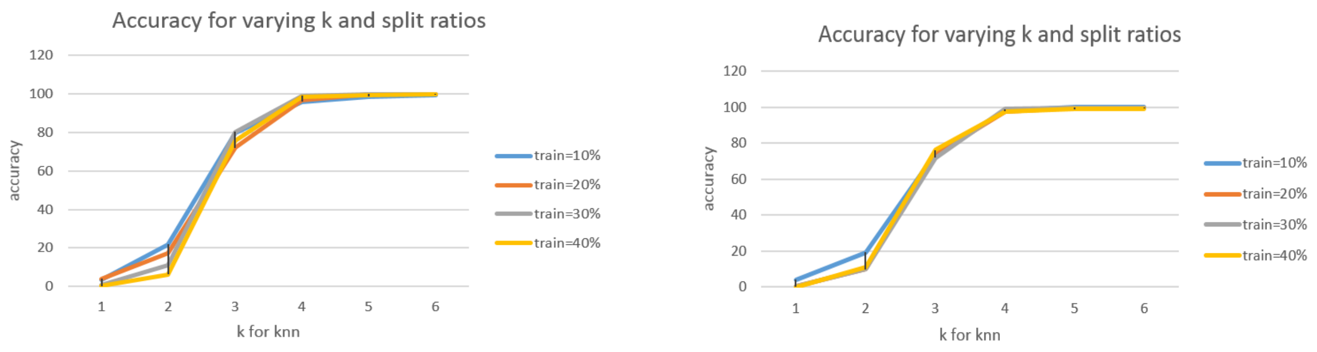


Figure 7 Accuracy versus the value of k in k nearest neighbor (for complete data set) for (a) Scheduled Caste and (b) Scheduled Tribe

4.4. Principal Component Analysis results

The dimensionality reduction achieved from PCA can be seen in the Figure 8 below for the Northern, southern and all states, respectively. The results are depicted for outlier with the value of 2 times the standard deviation added to the mean. The dashed line shows the threshold and the points above this threshold are considered as negative class data-points.

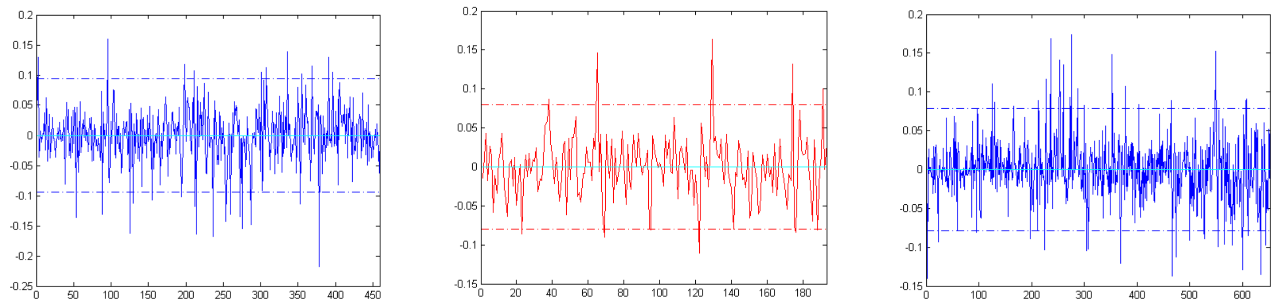
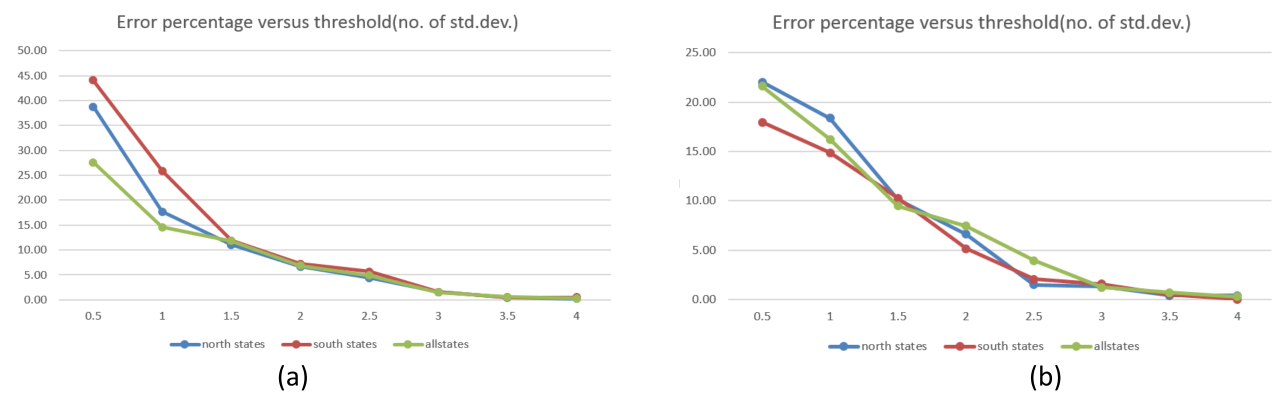


Figure 8 PCA results with respect to threshold equal to mean over 2 times the standard deviation (a) Northern States, (b) Southern States and (c) All states

Based on the results from the dimensionality reduction, for every set of data i.e. the northern states, southern states and all states, the error in the PCA is measured as the function of the outlier value. This can be seen in Figure 9 below. As can be seen from the figure, the error of the model decreases with increasing number of standard deviations as the outliers' threshold, which is very well expected. The scheduled castes' and scheduled tribes' results both show a very small gap between the error percentages, depicting that the systems are almost equally efficient for both the northern and southern states. For all states, for scheduled castes, the error looks to be lower compared to the north and south states. If two times standard deviation is considered for outlier, then for Scheduled caste, the error percentage for north, south and all states come around 7.5%, but for Scheduled tribes, it is in the range of 5-7.5% (7.5% for all states, 5% for south states and around 6% for north states). This shows that for both SC and ST, the results are almost similar for north, south and all states, and PCA doesn't distinguish between them significantly, and the reservation system is equally (in)efficient for all of them.



(a) (b)

Figure 9 Error percentage versus number of standard deviation above the mean value for PCA (a) Scheduled Caste and (b) Scheduled Tribe

4.5. Comparative results

For the first sub-research question, "*For the scheduled castes and tribes, does there exist a difference in the northern states and the southern states for efficiency of reservation system?*" the data is split into northern and southern states and northern states are taken as the train set, while southern as the test set. The results can be seen in table 3 column 3. SVM and kNN (for k<=3) show that the reservation system in the north and the south are different. ONB, KNN (k>3) and PCA shows the difference to be low, with PCA showing the smallest difference. Tribes have a higher difference for all the methods except SVM. This shows that there exists a difference in the reservation system for both the northern and southern states, especially for tribes.

For answering the second sub-research question: "*Are the socioeconomic conditions in the all states justifying the reservation system in India?*", and also verifying the above stated results, split tests are performed for SVM, ONB and KNN. For PCA, different values of standard deviation are tested for the complete dataset. The results can be seen in column 4 of table 3. All results (knn with k>3) show that the accuracy of the model is more than 75%, with highest accuracy for PCA, followed by KNN, SVM and ONB in decreasing order.

For answering the third sub-research question "*What can be an effective way to define reservation system in India for future based on the present data?*", the highest accuracy model would be suggested here, which also shows some difference in the north and south states, due to their differences in reservation allocation [28]. This would be the boundary method, k nearest neighbor, with k>3.

Table 3: Results of all the algorithms

1	2	3	4
		North-South (Training/Test set)	Split tests (except for PCA)
SVM (Boundary method)	SC	Different: 16% difference	Slightly justified (Error for 20% parameter: 15%-25%)
	ST	Different: 16% difference	
ONB (Density method)	SC	Similar: 5% max. difference	Slightly justified (Error for 20 percentile: 17%-27%)
	ST	Slightly Different: 10% max. difference	
KNN (Boundary method)	SC	Slightly Different: Difference 0-15%	Justified (0-25% for k=3 and 0-5% for k>3)
	ST	Different: Difference 0-25%	
PCA (Reconstruction method)	SC	Similar: 0% difference (2 time std. dev.)	Justified (Error for 2 times Std. dev.: 7.5%)
	ST	Similar: 0-2.5% diff. (2 time std. dev.)	

5. Conclusions and Reflection

Indian reservation system is a hot debate which questions the system’s efficiency. From a data analysis point of view, this is a question of predictive modelling as the class of whether a caste/tribe is labelled as “scheduled” is decided based on some attributes or demographic values. The research here aims at determining the same using census data of 2011 of India, which shows every castes’ demographics which are labelled as scheduled. Preprocessing of this data utilized excel macros, resulted in a total of 8 demographics as the indicators to decide the label for the caste as “scheduled” or not. As the non-scheduled caste data is not available, one class classification method is used for predictive modelling here by applying four different methods: one class support vector machine, one class Naïve Bayes, k nearest neighbor and principal component analysis.

The research question “*Is the reservation system for scheduled caste and scheduled tribe equally efficient for different states and is it well defined by the socio-economic indicators?*” is answered via three sub questions. Firstly, the boundary methods show that the difference in north-south is much larger than what is shown by density method, while the reconstruction method has the lowest difference (negligible). The reason behind this can be twofold; First, the reservation system in north and south is indeed different, as the quota (% of reservation) for some of the states is very high (as high as 69%), while this is not the case in north (till 2011, in 2014, some north states got quota above 50%) [28]. Second, the boundary method has lower accuracy here because they define the boundaries very clearly as compared to density methods and reconstruction method, where boundaries are based on a distribution and not exact distance as in KNN.

Secondly, the socioeconomic conditions justify the reservation systems with an accuracy of around 80% with boundary methods (in kNN, for k<=3), and around 78% with Density method and best with the reconstruction method with around 92.5% accuracy. Overall, the reservation system can be said to be not perfectly accurate, but not completely inefficient too.

Thirdly, kNN method with k>3, is the most suitable method here, because it does not only show the difference in north and south states to some extent, but also has higher accuracy (lower than PCA, but PCA shows no difference in north and south, which is not correct). This model would be most suitable for future predictions of the reserved castes in India, to make the reservation system efficient enough.

References

[1] Census India, "2011 census," 2012. [Online]. Available: http://www.censusindia.gov.in/2011census/population_enumeration.html. [Accessed 2016].

[2] Nitisha, "Reservation in India: Controversy, Justification and Controversy," Yourarticlecompany, 2015. [Online]. Available: http://www.yourarticlelibrary.com/reservation-system/reservation-in-india-controversy-justification-and-criticism/47114/. [Accessed July 2016].

[3] A. Bari, M. Chaouchi and T. Jung, "How to Choose an Algorithm for a Predictive Analysis Model," Predictive Analytics For Dummies, 2016. [Online]. Available: How to Choose an Algorithm for a Predictive Analysis Model. [Accessed 2016].

[4] J. Howard and M. Bowles, "The Two Most Important Algorithms in Predictive Modeling Today," Conferences Oreilly, 2012. [Online]. Available: http://conferences.oreilly.com/strata/strata2012/public/schedule/detail/22658. [Accessed 2016].

[5] GKollios. [Online]. Available: www.cs.bu.edu/fac/gkollios/ada01/LectNotes/Classification1.ppt.

[6] O. Mazhelis, "One-class classifiers: a review and analysis of suitability in the context of mobile-masquerader detection.," *Arima Journal 6,* pp. 29-48, 2007.

[7] Q. Wang, L. Lopes and D. Tax, "Visual object recognition through one-class learning," in *International Conference on Image Analysis and Recognition*, 2004.

[8] K. Wang and S. Stolfo, "One class training for masquerade detection," ICDM: Workshop on Data Mining for Computer Security , 2003.

[9] F. Denis, R. Gilleron and M. Tommasi, "Text classification from positive and unlabeled examples," in *9th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, 2002.

[10] B. Liu, W. Lee, P. Yu and X. Li, "Partially supervised classification of text documents.," in *Proc. of ICML*, 2002.

[11] A. L. N. R. D. Dempster, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society,* 1977.

[12] D. de Ridder, D. Tax and R. Duin, "An experimental comparison of one-class classification methods," *Proceedings of the 4th Annual Conference of the Advanced School for Computing and Imaging,* 1998.

[13] D. Munroe and M. Madden, "Multi-class and single-class classification approaches to vehicle model recognition from images," in *Irish Conference on Artificial Intelligence and Cognitive Science*, Portstewart , 2005.

[14] F. De Comite, F. Denis, R. Gillerson and F. Letouzey, "Positive and unlabeled examples help learning," *LNCS,* vol. 1720, p. 219–230, 1999.

[15] J. Quinlan, " Programs for Machine Learning," 1993.

[16] F. D. F. G. R. Letouzey, "Learning from positive and unlabeled examples," in *Algorithmic Learning Theory: 11th International Conference*, Sydney, 2000.

[17] Data School, "Comparing supervised learning algorithms," Data School, 2015. [Online]. Available: http://www.dataschool.io/comparing-supervised-learning-algorithms/.

[18] E. Chen, "Choosing a Machine Learning Classifier," 2011. [Online]. Available: http://blog.echen.me/2011/04/27/choosing-a-machine-learning-classifier/. [Accessed 2016].

[19] X. Amatriain, "What are the advantages of different classification algorithms?," Quora, [Online]. Available: What are the advantages of different classification algorithms?.

[20] R. Schapire, "Machine Learning Algorithms for Classification," Princeton University, 2015.

[21] D. H. &. W. R. Wolpert, "he relationship between PAC, the statistical physics framework, the Bayesian framework, and the VC framework," *In,* 1994.

[22] V. N. Vapnik, Statistical Learning Theory, Toronto: John Wiley & Sons, Inc, 1998.

[23] V. Kecman, "Support vector machines - an introduction," in *Support Vector Machines: Theory and Applications*, L. Wang, Ed., Berlin, Springer, 2005, pp. 1-47.

[24] S. S. Khan and M. G. Madden, "A Survey of Recent Trends in One Class Classification," in *Lecture Notes in Computer Science: Artificial Intelligence and Cognitive Science*, L. Coyle and J. Freyne, Eds., Dublin, Springer, 2009, pp. 188-197.

[25] B. Scholkopf, R. Williamson, A. Smola, J. Shawe-Taylor and J. Platt, "Support Vector Method for Novelty Detection," *Advances in Neural Information Processing Systems,* no. 12, pp. 582-588, 2000.

[26] G. Li, N. Japkowicz, I. Hoffman and R. K. Ungar, "Probability Calibration by The Minimum and Maximum Probability Scores in One-Class Bayes Learning for Anomaly Detection," *Conference on Intelligent Data Understanding,* pp. 160-174, 2010.

[27] G. J. N. H. I. &. U. R. K. Li, "Probability Calibration By the Minimum and Maximum Probability Scores in One-Class Bayes Learning for Anomaly Detection," *CIDU,* pp. 160-174, 2010.

[28] Ambedkar.org, "Ambedkar.org," 2011. [Online]. Available: http://www.ambedkar.org/News/reservationinindia.pdf. [Accessed July 2016].

[29] D. d. Ridder, D. M. Tax and R. P. Duin, "An experimental comparison of one-class classification methods," in *Proceedings of the 4th Annual Conference of the Advacned School for Computing and Imaging*, 1998.

[30] S. S. Khan and M. G. Madden, "A Survey of Recent Trends in One Class Classification," *Artificial Intelligence and Cognitive Science,* pp. 188-197, 2009.

[31] B. Rohner, "How to choose algorithms for Microsoft Azure Machine Learning," 2016. [Online]. Available: https://azure.microsoft.com/en-us/documentation/articles/machine-learning-algorithm-choice/. [Accessed 2016].

[32] J. Davis, "Bringing Predictive Modeling to Health Care," American Marketing Association, 2015. [Online]. Available: https://www.ama.org/publications/eNewsletters/MHSNewsletter/Pages/bringing-predictive-modeling-to-health-care.aspx#sthash.1zvdEG3f.dpuf. [Accessed 2016].

[33] D. Crockett, "4 Essential Lessons for Adopting Predictive Analytics in Healthcare," Health catalyst, 2016. [Online]. Available: https://www.healthcatalyst.com/predictive-analytics-healthcare-lessons/2/. [Accessed 2016].

[34] S. Loria, "Tutorial: Building a Text Classification System," Textblob, 2014. [Online]. Available: http://textblob.readthedocs.io/en/dev/classifiers.html. [Accessed 2016].

[35] Q. Wang and L. S. Lopes, "ONE-CLASS LEARNING FOR HUMAN-ROBOT INTERACTION," in *Emerging solutions for manufacturing systems*, Springer, 2005.

Appendix

Appendix 1: Pre-processing of data

Table 4 Description of the attributes used for the analysis in detail

Code	Description/ Title	Indicator extracted
SC – 05/ ST – 05	Marginal Workers And Non-Workers Seeking/ Available For Work Classified By Educational Level, Age And Sex	○ Illiterate Population / Total Population: Illiterate percentage in the caste ○ Literate people with primary education / Total literate population : Students in primary education per literate in the caste ○ Literate people with graduate and above education / Total literate population: Graduate and above educated per literate in the caste ○ Literate people with secondary or matric education / Total literate population : Secondary/ matric educated per literate in the caste
SC – 12/ ST – 12	Number Of Women And Ever Married Women By Present Age, Number Of Surviving Children And Total Surviving Children By Sex	○ Total surviving male per total surviving children: Surviving male children / Total surviving children ○ Total surviving children per total married women: Total surviving children / Total married woman
SC – 10/ ST – 10	Population Age 5-19 Attending School/College By Economic Activity Status And Sex	○ Total child labour / Total children population : Percentage of child-labor in the caste
A10/ A11	Individual Scheduled Caste Primary Abstract Data	○ Total working population / Total population : Total employment rate in the caste

Some of the removed parameters for SC and ST are (because they do not provide information on how progressed a caste is, but just gives descriptive data):

- Marital Status of the population
- Number Of Women And Ever Married Women By Present Age, Parity And Total Children Ever Born By Sex
- Number Of Women And Currently Married Women By Present Age, Number Of Births Last Year By Sex And Birth Order
- Population By Religious Community

Different states are divided based on their codes (which are given geographically) and the code till 27 is used as north, and from 27 onwards are used as southern state

Andhra Pradesh	28	Meghalaya	17	Karnataka	29	Tamil Nadu	33
Assam	18	Mizoram	15	Kerala	32	Tripura	16
Chhattisgarh	22	Orrisa	21	Madhya Pradesh	23	Uttarakhand	5
Delhi	7	Punjab	3	Manipur	14	Rajasthan	8
Goa	30	Uttar Pradesh	9	Haryana	6	Bihar	10
Gujarat	24	Maharashtra	27	Himachal Pradesh	2	West Bengal	19
Jammu Kashmir	1	Sikkim	11	Jharkhand	20		

Appendix 2: Choice of models

The different type of classification models for one class classification are explained below [29]:

1) Density estimation: Gaussian model, mixture of Gaussians and Parzen density estimators
    a. Straightforward method: to estimate the density of the training data and to set a threshold on this density
    b. Advantageous when: a good probability model is assumed; and the sample size is sufficient
    c. Rule of accepting: By construction, only the high density areas of the target distribution are included
    d. Gaussian density estimation:
        i. Insensitivity to scaling of the data: utilizing the complete covariance structure of the data
        ii. Another advantage: computing the optimal threshold for a given true positive
    e. Parzen density estimation: Cheap training cost, but expensive testing cost: all training objects have to be stored and distances to all training objects have to be calculated and sorted
2) Boundary methods: k-centers, **NN-d and SVDD**
    a. NN-d
        i. Advantages: avoids density estimation and only uses distances to the first nearest neighbor
        ii. Local density is estimated by:

$$p_{\mathrm{NN}}(\mathbf{z}) = \frac{k/N}{V_k(\|\mathbf{z} - \mathrm{NN}_k^{tr}(\mathbf{z})\|)}$$

        iii. a test object z is accepted when: its local density is larger or equal to the local density of its nearest neighbor in the training set
    b. SVM: To minimize structural error: $\mathcal{E}_{\mathrm{struct}}(R, \mathbf{a}) = R^2$ with the constraints: $\|\mathbf{x}_i - \mathbf{a}\|^2 \leq R^2, \quad \forall i$
3) Reconstruction methods: k-mean clustering, self-organizing maps, **PCA** and mixtures of PCA's and diabolo networks
    a. Most of the methods make assumptions about the clustering characteristics of the data or their distribution in subspaces
    b. A set of prototypes or subspaces is defined and a reconstruction error is minimized
    c. Differs in: definition of prototypes or subspaces, reconstruction error and optimization routine
    d. PCA:
        i. Used for data distributed in a linear subspace
        ii. Finds the orthonormal subspace which captures the variance in the data as best as possible
        iii. To minimize the square distance from the original object and its mapped version:

$$d_{PCA}(\mathbf{z}) = \left\|\mathbf{z} - (\mathbf{W}(\mathbf{W}^T\mathbf{W})^{-1}\mathbf{W}^T)\mathbf{z}\right\|^2 = \left\|\mathbf{z} - (\mathbf{W}\mathbf{W}^T)\mathbf{z}\right\|^2$$

    e. Kernel PCA
        i. Can efficiently compute principal components in high-dimensional feature spaces, related to input space by some nonlinear map
        ii. Indistinguishable problems in original spaces can be distinguished in mapped feature space with the map $\Phi : R^N \to F, x \mapsto X.$
        iii. The map need not to be obviously defined because of inner products can be reduced to kernel functions

Appendix 3: Codes for the algorithms

App 3.1. Code for SVM

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.font_manager
from sklearn import svm
import csv
import random
from __future__ import division

%matplotlib inline

def loadCsv(filename):
   lines = csv.reader(open(filename, "rb"))
   dataset = list(lines)
   for i in range(len(dataset)):
      dataset[i] = [float(x) for x in dataset[i]]
   return dataset

def splitDataset(dataset, splitRatio):
   trainSize = int(len(dataset) * splitRatio)
   trainSet = []
   copy = list(dataset)
   while len(trainSet) < trainSize:
      index = random.randrange(len(copy))
      trainSet.append(copy.pop(index))
   return [trainSet, copy]
####### SCHEDULED CASTE #######
filename = 'SC_import_osvm.csv'
dataset = loadCsv(filename)
colors = ['k', 'r', 'g', 'c', 'y', 'm', 'b', 'purple', 'lime', 'dimgray']
for j in range(0,9):
    splitRatio = 0.8
    trainingSet, testSet = splitDataset(dataset, splitRatio)

    #fit the model while modifying nu parameter
    #set sensitivity test for threshold between 0-20%
    nuparameter = np.arange(0.01, 0.21, 0.01)
    percentage_error_train_list = []
    percentage_error_test_list = []
    globals()['testdefect{}'.format(j)] = []
    for i in nuparameter:
        clf = svm.OneClassSVM(nu= i , kernel="rbf")
        clf.fit(trainingSet)
        y_pred_train = clf.predict(trainingSet)
        y_pred_test = clf.predict(testSet)
        n_error_train = y_pred_train[y_pred_train == -1].size
        n_error_test = y_pred_test[y_pred_test == -1].size
        percentage_error_train = n_error_train / y_pred_train.size
        percentage_error_test = n_error_test / y_pred_test.size
        percentage_error_train_list.append(percentage_error_train)
        percentage_error_test_list.append(percentage_error_test)
    globals()['plotdata{}'.format(j)] = plt.plot(nuparameter,percentage_error_test_list,
color = colors[j])


plt.xlabel('Nu Parameter', fontsize=16)
plt.ylabel('Percentage Error', fontsize=16)
plt.show()
```

App 3.2. Code for ONB

```
#FUNCTION DEFINITION

import csv
import random
import math
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline

def loadCsv(filename):
   lines = csv.reader(open(filename, "rb"))
   dataset = list(lines)
   for i in range(len(dataset)):
      dataset[i] = [float(x) for x in dataset[i]]
   return dataset

def splitDataset(dataset, splitRatio):
   trainSize = int(len(dataset) * splitRatio)
   trainSet = []
   copy = list(dataset)
   while len(trainSet) < trainSize:
      index = random.randrange(len(copy))
      trainSet.append(copy.pop(index))
   return [trainSet, copy]

def separateByClass(dataset):
   separated = {}
   for i in range(len(dataset)):
      vector = dataset[i]
      if (vector[-1] not in separated):
         separated[vector[-1]] = []
      separated[vector[-1]].append(vector)
   return separated

def mean(numbers):
   return sum(numbers)/float(len(numbers))

def stdev(numbers):
   avg = mean(numbers)
   variance = sum([pow(x-avg,2) for x in numbers])/float(len(numbers)-1)
   return math.sqrt(variance)

def summarize(dataset):
   summaries = [(mean(attribute), stdev(attribute)) for attribute in zip(*dataset)]
   del summaries[-1]
   return summaries

def summarizeByClass(dataset):
   separated = separateByClass(dataset)
   summaries = {}
   for classValue, instances in separated.iteritems():
      summaries[classValue] = summarize(instances)
   return summaries

def calculateProbability(x, mean, stdev):
   exponent = math.exp(-(math.pow(x-mean,2)/(2*math.pow(stdev,2))))
   return (1 / (math.sqrt(2*math.pi) * stdev)) * exponent

def calculateClassProbabilities(summaries, inputVector):
   probabilities = {}
   for classValue, classSummaries in summaries.iteritems():
      probabilities[classValue] = 1
      for i in range(len(classSummaries)):
         mean, stdev = classSummaries[i]
         x = inputVector[i]
         probabilities[classValue] *= calculateProbability(x, mean, stdev)
   return probabilities


def calculateClassProbability(summaries, inputVector):
   probabilities = {}
   for classValue, classSummaries in summaries.iteritems():
      probabilities[classValue] = 1
      for i in range(len(classSummaries)):
         mean, stdev = classSummaries[i]
         x = inputVector[i]
         probabilities= calculateProbability(x, mean, stdev)
   return probabilities


def getProbabilities(summaries, testSet):
   predictions = []
   for i in range(len(testSet)):
      result = calculateClassProbability(summaries, testSet[i])
      predictions.append(result)
   return predictions

def predict(summaries, inputVector):
   probabilities = calculateClassProbabilities(summaries, inputVector)
   bestLabel, bestProb = None, -1
   for classValue, probability in probabilities.iteritems():
      if bestLabel is None or probability > bestProb:
         bestProb = probability
         bestLabel = classValue
   return bestLabel

def getPredictions(summaries, testSet):
   predictions = []
   for i in range(len(testSet)):
      result = predict(summaries, testSet[i])
      predictions.append(result)
   return predictions

def getAccuracy(testSet, predictions):
   correct = 0
   for i in range(len(testSet)):
      if testSet[i][-1] == predictions[i]:
         correct += 1
   return (correct/float(len(testSet))) * 100.0

def calculateOutlier(testprobabilities):
    positiveclass = []
    for i in range(len(testprobabilities)) :
        if testprobabilities[i] > threshold:
            positiveclass.append(1)
        else:
            positiveclass.append(0)
    positivepercentage = float(sum(positiveclass)) / len(positiveclass)
    return positivepercentage

# prepare model
filenorth = 'SC_import_north_NB.csv'
trainingSet = loadCsv(filenorth)

filesouth = 'SC_import_south_NB.csv'
```

```
testSet = loadCsv(filesouth)

#summary (mean, std dev) of each attribute in the training set
summaries = summarizeByClass(trainingSet)
#Bayesian Probabilities of the Training Set
initialprobabilities = getProbabilities(summaries, trainingSet)
threshold = min(initialprobabilities)
#Calculate the percentage of the test set which is considered novel/outlier from the training
set
testprobabilities = getProbabilities(summaries, testSet)
calculateOutlier(testprobabilities)
#set sensitivity test for threshold between 0-20%
thresholdpercentage = range(0, 21)

positivepercentagelist = []
for i in thresholdpercentage:
    threshold = np.percentile(initialprobabilities, i)  # return ith percentile
    positivepercentage = calculateOutlier(testprobabilities) * 100
    positivepercentagelist.append(positivepercentage)

plt.scatter(thresholdpercentage, positivepercentagelist)
plt.xlabel('Threshold Percentile', fontsize=16)
plt.ylabel('Positive Value Percentage', fontsize=16)
plt.show()
```

App 3.3. Code for KNN

```
import numpy as np
import csv
import matplotlib.pyplot as plt
import random
import sys
with open('Y_all.csv') as f:
    n=740
    er=[]
    kn=[]
    reader = csv.reader(f, delimiter=',')
    data = [(float(col1), float(col2), float(col3), float(col4), float(col5), float(col6),
float(col7), float(col8))
                for col1, col2, col3, col4, col5, col6, col7, col8 in reader]
    for gg in range(10,50,10):
        datad=[]
        datac=[]
        length=int((gg*len(data))/100)
        for i in range (0,length-1,1):
            datad.append(data[i])
        for i in range (length,len(data)-1,1):
            datac.append(data[i])
        for knn in range (1,7,1):
            if knn==1:
                cc=2
            elif knn==2:
                cc=3.14
            elif knn==3:
                cc=4.19
            elif knn==4:
                cc=4.935
            elif knn==5:
                cc=5.264
            elif knn==6:
                cc=5.168
            print knn
            s1=[]
            for i in range(0,length-1,1):
                    d=[]
                    for k in range (0,length-1,1):
                            x=[]
                            if(k!=i):
                                for j in range (0,7,1):
                                    x.append((datad[i][j]-datad[k][j])**2)
                                y=np.sum(x)
                                sqsum=np.sqrt(y)
                                d.append(sqsum)
                    m=min(d)
                    local_density=knn/((n*cc)*(m**knn))
                    s=[i,m, d.index(m),local_density]
                    s1.append(s)
            #now check from here if the testing set has its local density larger than the
local density of the above training set
            g=[]
            u=[]
            v1=[]
            for h in range (0,len(data)-length-1,1):
                d=[]
                for w in range (0,length-1,1):
                    x=[]
                    if(w!=h):
```

```
                for j in range (0,7,1):
                    x.append((datac[h][j]-datad[w][j])**2)
                y=np.sum(x)
                sqsum=np.sqrt(y)
                d.append(sqsum)
        m=min(d)
        local_density=1/(651*(4/3)*(3.14*m**3))
        v=[h,m, d.index(m),local_density]
        v1.append(v)
    c=[]
    for p in range (0,len(v1),1):
        for o in range (0,len(s1)-1,1):
            if v1[p][2]==s1[o][0]:
                if v1[p][3]>s1[o][3]:
                    c.append(p)
    acc=float(len(c))/(len(data)-length-1)
    err= (1-acc)*100
    print err
    er.append(err)
    kn.append(knn)
plt.plot(kn,er,marker='o', linestyle='--', color='r')
plt.xlabel('k for knn', fontsize=16)
plt.ylabel('Accuracy', fontsize=16)
plt.show()
```

App 3.4. Code for PCA

```
clc
clear
load('Y_all.csv', 'Y_all')
n=740;
m=8;
Y_all_norm = zscore(Y_all);%normalize Y_all
[V,D] = eig(Y_all_norm'*Y_all_norm,'balance');%V eigenvector (descend), D
eigenvalue (descend)-lambda
[D_descend,indeY_all] = sort(D,'descend');
V_descend = [];
for i = 1:size(indeY_all,1)
    V_descend = [V_descend,V(:,indeY_all(i))];
end

eigenflow = zeros(size(Y_all_norm(:,1)));
sum = 0;
for i = 1:size(D_descend)
    %eigenflow(:,i) = (Y_all*V_descend(:,i));%U vector
    eigenflow(:,i) = (Y_all_norm*V_descend(:,i))/norm((Y_all_norm*V_descend(:,i)));
end
break_inside = 0;
for i = 1:m
    for j = 1:n
        if (eigenflow(j,i) > mean(eigenflow(:,i))+1*std(eigenflow(:,i))) ||
(eigenflow(j,i) < mean(eigenflow(:,i))-1*std(eigenflow(:,i)))
            norm_num = i;%which u vector
            norm_time = j;%which time point
            break_inside = 1;
            break
        end
    end
    if break_inside == 1
        break
    end
end
figure(1)
plot(eigenflow(:,norm_num),'b-')
hold on
plot([0,n],[mean(eigenflow(:,norm_num)),mean(eigenflow(:,norm_num))],'c-')
plot([0,n],[mean(eigenflow(:,norm_num))+1*std(eigenflow(:,norm_num)),mean(eigenflow
(:,norm_num))+1*std(eigenflow(:,norm_num))],'b-.')
plot([0,n],[mean(eigenflow(:,norm_num))-
1*std(eigenflow(:,norm_num)),mean(eigenflow(:,norm_num))-
1*std(eigenflow(:,norm_num))],'b-.')
xlim([0 n])
```