Minimizing Dataset Size Requirements for Machine Learning

by

Salil Batra

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved April 2017 by the
Graduate Supervisory Committee:

John Femiani, Co-Chair
Ashish Amresh, Co-Chair
Ajay Bansal

ARIZONA STATE UNIVERSITY

May 2017

ABSTRACT

Machine learning methodologies are widely used in almost all aspects of software engineering. An effective machine learning model requires large amounts of data to achieve high accuracy. The data used for classification is mostly labeled, which is difficult to obtain. The dataset requires both high costs and effort to accurately label the data into different classes. With abundance of data, it becomes necessary that all the data should be labeled for its proper utilization and this work focuses on reducing the labeling effort for large dataset. The thesis presents a comparison of different classifiers performance to test if small set of labeled data can be utilized to build accurate models for high prediction rate. The use of small dataset for classification is then extended to active machine learning methodology where, first a one class classifier will predict the outliers in the data and then the outlier samples are added to a training set for support vector machine classifier for labeling the unlabeled data. The labeling of dataset can be scaled up to avoid manual labeling and building more robust machine learning methodologies.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Machine learning has now become an essential part of many software applications. In its traditional application, classification models require large sets of inputs labeled data. Manual data labelling at large scale becomes highly expensive in terms of both cost and time. This thesis work focuses on how we can minimize the amount of labeled data to train a machine effectively and still produce accurate results.

Learning from small dataset is a challenging problem and many methodologies try to achieve high accuracy with large datasets, for example, One Class Classifiers (OCC), Semi-Supervised Learning (SSL) / Online Machine Learning (OML) methodologies. OCC as instance, uses only positive samples or positive with very little negative samples to train, however, using only one class still requires labelling all the positive samples. The traditional classification problem uses both positive and negative datasets and provide more accurate results given that the data is labeled accurately.

Labeling a large dataset is a cumbersome task, however, unlabeled data is available in abundance. Utilizing a small label dataset to label unlabeled data, an active learning methodology, can be utilized to reduce the problem space. The drawback in such scenario is to either use an oracle (eg. a human to label all the data piece by piece acquired by querying the classifier to provide a prospective label) or depend on a classifier to accurately label the unlabeled pool. Both the scenarios carry disadvantages and the assumptions that all data is true which is, the data is balanced, no sample is wrongly labeled, no parts in the dataset is missing. Many techniques such as normalizing data, taking Gaussian for the entire data, and others can be used to address true data problems which contributes to higher accuracy of a classification model but does not completely address reducing the dataset size requirements.

Many large datasets with large feature space can be reduced to fewer features (dimensionality reduction) using techniques such as Principal Component Analysis (PCA). The new reduced dimensionality can be thought of less data in feature space, however, the classifier still trains and tests on same number of samples. PCA can address hard problems of classification, such as curse of dimensionality, however could lead to loss in important data for less dimensions.

Object detection in geo-spatial images is a well-known problem and has been solved by first labeling huge datasets for positive and negative samples, and then classifying the datasets with traditional models such as State Vector Machine (SVM), Active Learning methodology, and others. The accuracy of such methodologies is generally high because of the decision boundary being very vast and large amount of both negative and positive samples provide higher accuracy.

OCC can help in lowering the required labeled data for higher accuracy with requiring only positive or positive with little negative samples to train the classifier. This methodology helps in reducing the dataset size required for training, however still carries disadvantages such as curse of dimensionality, inlier and outlier detection, which become much harder due to the absence of negative data. The OCC is discussed in detail in chapter 2.

This work tries to find the smallest size to train a classifier and compare accuracy results. The comparison is drawn between multi-class classifiers and OCC for both traditional machine learning and active learning methodology.

OCC has also been adopted for active learning methodology. In the work as presented by Barnab-Lortie, Bellinger, & Japkowicz (2015), methods for OCC such as- Mahalanobis distance classifier, distance to KNN, and One-Class SVM can be used to devise an active learning model. The author proposes different set of procedures to compare how OCC with KNN and Mahalanobis distance and One-Class SVM performs. The idea presented is that, first a small dataset is used to train OCC and a little data is kept for validation. Once the OCC is train and tested, use a selection criterion to choose samples from unlabeled pool. The samples are then used added to the original training set and tested on "held out" data.

In comparison to the work presented in this thesis, Barnab-Lortie, Bellinger, & Japkowicz (2015) do not focus on lowering the dataset size or reducing data preprocessing, but evalute performance of different OCC techniques. In this thesis work, we try to reduce the dataset size requirements by evaluating performance of classifiers based on small dataset without using data preprocessing to boost the performance. Also, we introduce a novel methodology to build an active machine learning model by combing One-Class SVM and binary SVM to reduce the data labeling cost and build a robust active learning model.

1.1 Motivation

For an application to effectively use machine learning models, it is an expensive task to collect and label large amounts of data. If we can reduce the labeled data and still predict the outcome with high accuracy, the overall costs can be largely reduced. With higher accuracy of a classifier, an active learning approach to label unlabeled dataset can also minimize labeling of data through an oracle.

With the progress in technology, all major software has data centric operation. The data however, might not be labeled into classes and may require manual effort to do so. The machine learning algorithms achieve high accuracy when trained with large labeled dataset size. Labeled dataset being expensive, some part of it must be kept for testing the performance of a classifier, which means some percentage of data is not utilized to make the model more accurate.

Another disadvantage with large labeled dataset is that the data pre-processing costs also increases. The training data should be kept true, which means, low on variance and missing values along with its distribution in standard, normalized, or Gaussian form.

The pre-processing costs with large datasets can be reduced if we can assert that small labeled dataset size is effective to achieve high accuracy.

1.2 Hypothesis

One Class Classification methods train on positive samples with no or limited negative samples. The limited negative samples can be either an inadequate small amount of labeled negative samples, poorly distributed negative samples or unlabeled data to extend the classification boundary. Filtering our data set with positive samples reduces the original dataset size and reduce the required manual labelling of the complete data set. With this we expect to see high accuracy when training with one class classifier for large positive sample data set. A multi-class classifier for the same amount of data might not perform as good as OCC due to small training size.

Another interesting viewpoint is then to check - how multi-class classifiers work with increasing amounts of input data in accuracy metric versus OCC. This will then lead to calculate precision and recall determining at what point the multi-class classifiers takes over one class classifier in terms of accuracy. We then extend this incremental approach to active learning methodology where, we first train a multiclass classifier with labeled data and random samples from unlabeled pool to determine if unlabeled data was accurately classified or not. Another extension is to train a OCC to detect inliers and outliers in the data and use those samples with labeled pool along with training data to check if unlabeled pool is accurately labeled. Techniques such as Random Over Sampling (ROS) (Lemaˆıtre, Nogueira, & Aridas, 2017) can be used to balance imbalances in data and remove under sampling problem. ROS will help in increasing the accuracy of the model and make the model robust.

CHAPTER 2

LITERATURE REVIEW

In this chapter, we will discuss One Class Classification (OCC), Active Learning, Support Vector Machines (SVM), and Principal Component Analysis (PCA).

2.1 One Class Classification

2.1.1 General Introduction to One Class Classification

In conventional classification techniques, we train our classifier using both positive and negative labeled data. Using such data helps in making the decision boundary cover most of the data instances, in other words it provides good generalization of the overall data. One Class Classification (OCC) on the other hand deals only with the following - only positive labels or positive with little negative samples. The goal here is to define a decision boundary such that all the data from positive samples is included and best matches from unlabeled data are accurately predicted. This provides an advantage over using less data and more so labeling only positive samples, reducing the manual labeling effort.

However, OCC still carries disadvantages such as - outlier detection, curse of dimensionality, and overfitting of data, as seen in a multi-class classifier. Lesser the data available for OCC classifier, the harder it becomes to achieve higher accuracy.

A OCC methodology deals in classifying objects with only positive samples or very little negative and positive sample dataset. The OCC has been widely studied and many comparisons have been drawn to distinguish the ease of usability of OCC vs multi-class setup and to highlight that both classification techniques have similar problems such as -

curse of dimensionality, boundary definition, estimation of classification error, etc. The OCC is also thought to be a difficult classification methodology due to lack of negative samples in the dataset.

One of the most common examples studied to compare one class with multi-class classification is, distinguishing fruits with vegetables. A positive only dataset will classify different fruits and vegetables based on the category they belong to. However, this problem becomes unrealistic when a test data such as an animal (let us consider a dog in this example) is fed to the classifier. The dog in this case will certainly be classified as either a fruit or a vegetable, which is wrong in both the cases.

Another example can be thought of while discussing OCC vs multiclass classification is detection of false sensor data. While collecting, and labeling a dataset, we would always have positive faults dataset or positive with very little negative sample dataset. In such a scenario, it becomes highly difficult to detect when the sensor is showing an abnormal/false behavior due to absence/little availability of negative samples, thereby making it harder to decide feature vectors for such data.

OCC problem as presented by Mazhelis is as follows:

Let us consider an object Z represented by a vector $x = (x_1, \dots, x_n)$ of values of n_f features from feature space. Let C be classes and C_i, where $i = \{1, \dots, N_C\}$, denotes label of class i.

Let DS_T and DS_C be training and test datasets where:

$DS_T = \{\left(\left(x_1, \dots, x_{nf}\right)_J, \ y_i | j = 1, \dots DS_T|\right\}$ y_i, is the class label, and

$DS_C = \{\left(\left(x_1, \dots, x_{nf}\right)_J, \ y_i | j = 1, \dots DS_C|\right\}$ are vectors of features without class labels.

Let θ be a set of parameters to be learned by the classifier. For an unlabeled observation x, the classifier should produce an output u (x, θ) where $u(x,\theta) \in \{C_1, \dots, C_{NC})$. The classifier can also implement a real-valued discriminant function $u_{ci}(x,\theta)$ such that it greater values of the function corresponds to higher probability of class membership $P(Z \in C_i \,|x) = P(C_i \,|x)$. Consequently, the highest value of discriminant function is selected:

$$\gamma(x,\theta) = argmax_{i=1,\dots,N_C} \, u_{c_i}(x,\theta),$$

where γ is a mapping function.

A highest posterior probability can be calculated using Baye's rule for any output of a classifier's approximated probability $P(C_i \,|x)$.

A single discriminant function for a two-class classification problem can be given as: $u(x,\theta) = \; P(C_1 \,|x) - \; P(C_2 \,|x)$ which can lead to implement classification as:

$\gamma(x,\theta) = \{C_1, if \; u(x,\theta) \geq 0, \; C_2 \, if \; u(x,\theta) < 0$

2.1.2 Putting to test- Performance of OCC

The performance of a multiclass classifier is computed based on probability density of both positive and negative classes. However, due to availability of only positive sample in OCC, the probability density of only positive class is known. The performance in OCC is therefore calculated by measuring objects - true positive, false positive, false negative, true negative. Table 1 is a confusion matrix which illustrates different scenarios.

	Object from target class	**Object from outlier class**
Classified as a target object	True positive, T^+	False positive, F^+
Classified as an outlier object	False negative, F^-	True negative, T^-

Table 1: A confusion matrix for One Class Classification. Source (Tax D. M., 2001)

From table 1, it can be deduced that $T^+ + F^- = 1$ and $F^+ + T^- = 1$. The complication in this scenario is that, nothing or very little is known about $F^+ \, and \, T^-$, however, T^+ and F^- can be estimated. Now to calculate the true error (ε_{true}), as discussed earlier, the complete probability density $p(x, y)$ should be known.

In case of OCC however, only $p(x|\omega_T)$ (ω_T being the target class) is known and only error of first kind ε1can be estimated. ε_1 being the reduction of false negatives detected in the matrix above. Error of second kind ε_{11} is the false positives from above matrix. This error is unmeasurable due to the absence of a negative class and hence an outlier distribution $p(x|\omega_O)$ cannot be estimated.

The posterior probability in this case can be calculated using Baye's rule:

$$p(\omega_T|x) = p(x|\omega_T)p(\omega_T)p(x)$$

$$= p(x|\omega_T)p(\omega_T)p(x|\omega_T)p(\omega_T) + p(x|\omega_O)p(\omega_O)$$

The outlier distribution $p(x|\omega_O)$ is assumed to be uniformly distributed and $p(x|\omega_O)$ is independent of x in feature space, which means $p(x|\omega_T)$ can be used instead of $p(\omega_T|x)$ transformed by a strictly increasing function.

2.1.3 Taxonomy of One Class Classification

Unified approach to OCC based on the prior work is proposed by (Khan & Madden, 2004). The approach presented is broadly divided into three categories listed as follows:

1. Availability of Training Data: Learning with positive only / positive and very little negative / unlabeled data.
2. Methodology Used: Algorithms based on One-class Support Vector Machines (OSVMs) or methodologies based on algorithms other than OSVMs.
3. Application Domain: Application of OCC in domains like image classification, text/document classification, or in other domains.

Let us now discuss the above taxonomy briefly.

2.1.3.1 Availability of Training Data.

The availability of training data is a key factor while discussing OCC. The training data which is large positive, labeled data plays a key role in not only generalizing the classifier but also, defining an efficient decision boundary. The outlier class, as in this case, should be clearly distinguished to avoid anomaly and detect false positive objects.

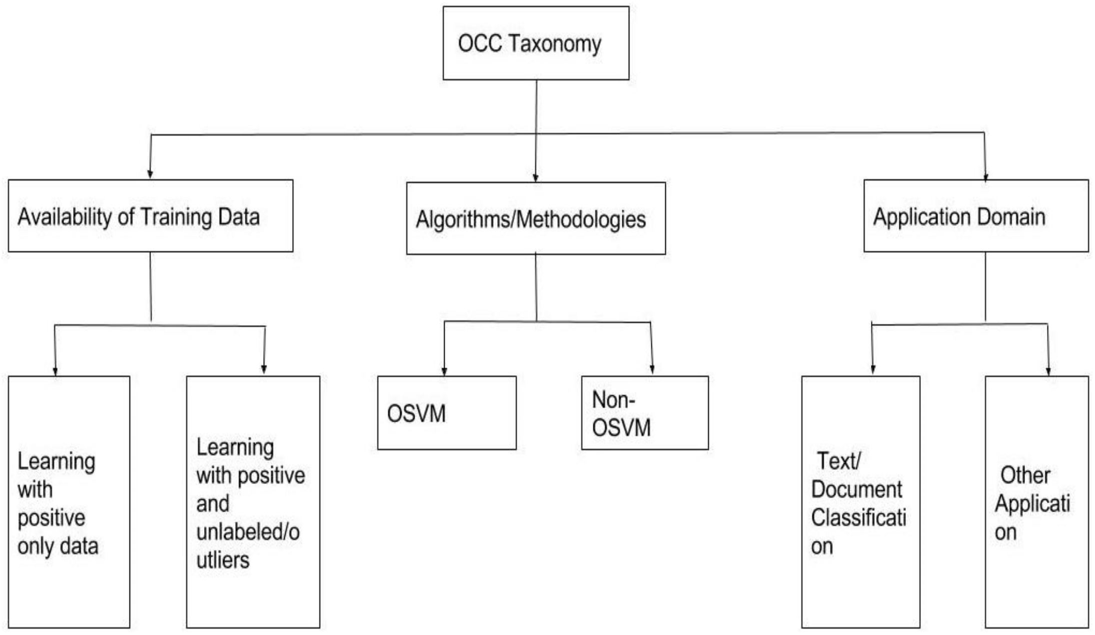
As discussed earlier, the training of OCC can be categorized in three categories:

1 Learning with positive only data.

2 Learning with positive and small negative set / artificial outliers.

3 Learning with positive and unlabeled data.

The above categories represent the type of data which can be used to train a OCC. The most common being the 1 and 2 have been thoroughly discussed in different research work. The third category, positive and unlabeled data, overlaps slightly with semi-supervised learning which is explained in following section.


OCC Taxonomy
Availability of Training Data
Algorithms/Methodologies
Application Domain
Learning with positive only data
Learning with positive and unlabeled/outliers
OSVM
Non-OSVM
Text/ Document Classification
Other Application
Figure 1: Taxonomy of OCC. Source: (Khan & Madden, 2004)

2.1.4 Methodology Used

Majority of methodologies involving OCC use One-class Support Vector Machines (OSVMs) or Non-OSVMs (neural networks, decision trees, nearest neighbors and many others). Let us discuss both the techniques in brief detail.

2.1.4.1 OSVMs

The study of OSVM involves the Support Vector Data Descriptors (SVDDs, detailed discussion in following section). The SVDDs comprises of a hyper-sphere / hyper-plane around the positive class data containing almost all the data points and at minimum radius. A Gaussian or Polynomial Kernel is typically implemented to compute a hyper-sphere / hyper-plane to manage higher dimensionality of feature space and define Support Vectors efficiently.

2.1.4.2 Methods other than OSVMs

Methods such as neural networks, nearest neighbors, decision trees, and others can also be used to classify objects in One Class. These methods have been widely studied and comparisons have been drawn with OSVM methodology. The study of these methods is out of scope for this work and are not discussed in detail.

2.1.4.3 Application Domain.

The application of OCC methodologies can be commonly seen in text/document classification problems. The other applications include handwriting detection, remote sensing, object recognition, spam detection, and many others.

2.1.5 OCC - Learning and Methodologies.

In this section, we will discuss learning and methodologies which can be utilized to build a OCC classifier as described by (Khan & Madden, 2004).

2.1.5.1 Training a Classifier

Before a classifier predicts objects or predicts the outcome we train the classifier on probability density function (PDF) or on parameters of discriminant functions.

Methods such as parametric or nonparametric density estimation can be used to determine PDF.

Assuming we have a Gaussian distribution $\theta = \{\mu, \Sigma\}$ parameters of the distribution can be estimated through maximizing the likelihood in form:

$$L(\mu, \Sigma) = \prod_{i \in DS_t} \quad p(x_i \,|\mu, \Sigma)$$

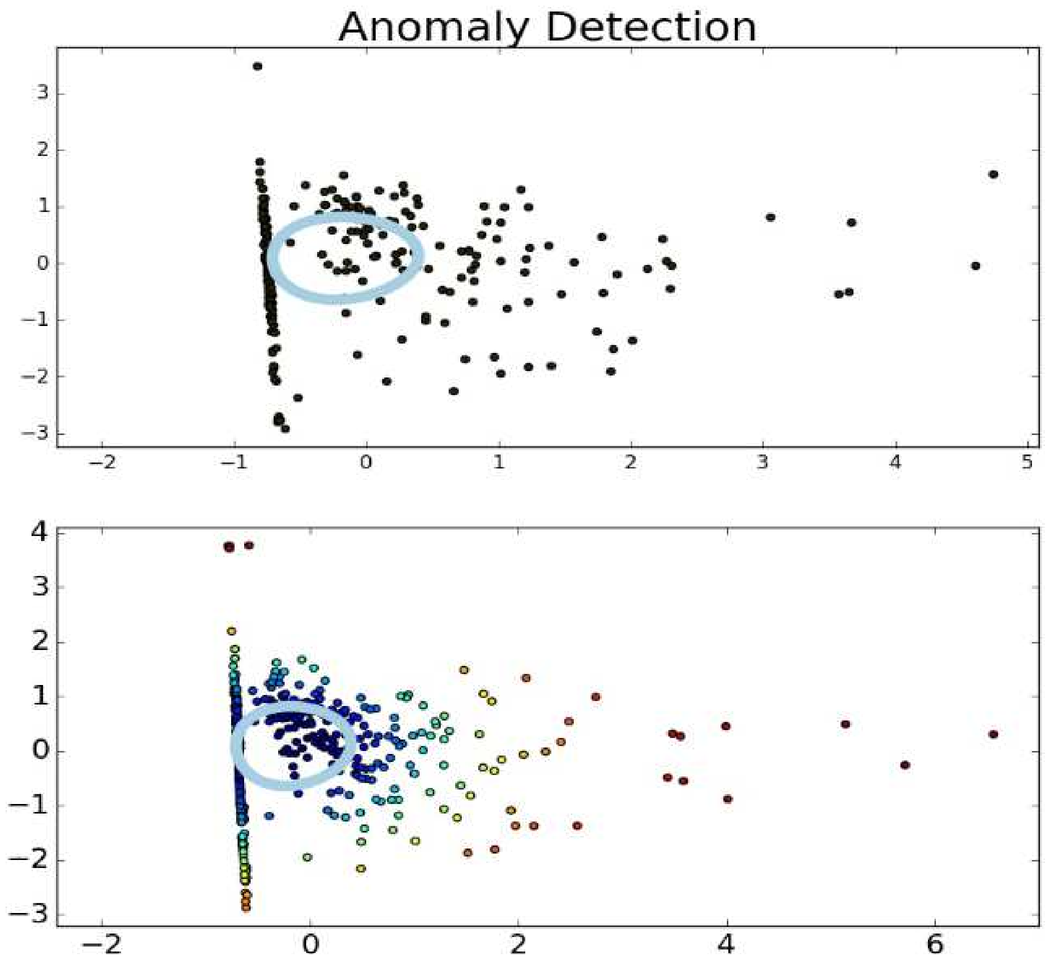


Figure 2: A sample anomaly detection curve from the dataset used in experiment. The blue circle shows the decision boundary of the classifier.

When it is difficult to estimate likelihood maximization, Expected-Maximization (EM) algorithm can be used to compute expectation and maximization. In expectation procedure (E-step) the expectation E of the log-likelihood of parameters with respect to hidden parameters is calculated. In maximization procedure (M-step) the parameters are reassigned the values maximizing the expectation of log-likelihood. The E and M steps are iteratively repeated until convergence.

Parameters of discriminant function can be estimated by minimizing an error function of parameters. The value of error function is evaluated using empirical data and it reflects degree of misclassification error. For instance, the cross-entropy error function is given by:

$$\varepsilon_{CE} = -\sum_{x_i \in DS_t} \sum_{k=1}^{2} y_{ik} \; ln\, u_k \,(x_i, \theta) \begin{cases} y_{ik} = 1, \text{if } x_i \in C_k \,, \\ y_{ik} = 0, \text{ otherwise} \end{cases}$$

Another concept used while defining learning of a classifier is *Generalizability*. The term generalizability refers to the ability of a classifier to include data points which lie outside the decision boundary making the classifier general enough and able to achieve high accuracy. Consider decomposition of misclassification error into bias and variance. The decomposition is expected error over all possible training set instead of just DS_T (training dataset). Let us take sum-of-square error function for regression problem instead of classification:

$$E_{DS_T}[(u(x_i, \theta) - y_i^2]$$

$$= \left\{E_{DS_T}[u(xi, \theta)] - yi\right\}^2 + E_{DS_T}[\{(u(xi, \theta)] - E_{DS_T}[(u(xi, \theta)\}^2]$$

$$= (bias)^2 + variance$$

where $y(x)$ is the regression function approximated. The bias tells us how flexible our model is and variance value defines the ability of a classifier to generalize beyond training dataset.

2.1.5.2 Classification Methodologies

In this section, we will discuss different methodologies which can be used to classify a OCC.

2.1.5.2.1 Histogram of Oriented Gradients (HoG)

The work of (Dalal & Triggs, 2005) has been widely recognized in human detection problem. The proposed work moves forward from previous work like HaaR feature extraction, for face recognition problems. The previously studied HaaR feature extraction method lacked in defining good feature vector. The work presented by (Dalal & Triggs, 2005) combines Scale Invariant Feature Transform (SIFT) feature extraction methodology with histograms to define comprehensive feature vectors and improve the accuracy. The HoG method is robust to the noise in training data along with mislabeling errors and depends on how the feature space is divided into bins. Gaussian and mixture of Gaussians methodology is also widely used to build an effective classifier.

Assuming a dataset is normally distributed, then the Gaussian distribution is given by:

$$p_G(x, \mu, \sum) = \frac{1}{(2\pi)^{\frac{d}{2}}|\sum|\text{^}1/2} \times \exp\{-\frac{1}{2}(x-\mu)^T \sum^{-1}(x-\mu)\}$$

where μ is the mean vector, and ∑ is covariance matrix.

Additionally, a mixture of Gaussian distribution represents linear combination n_{MG} as:

$$p_{MG}(x) = 1/nM_G \sum_{i=1}^{n_{MG}} p_G(x, \mu\_i, \sum\_i)P(i)$$

The number of parameters for Gaussian model is given by:

$$n_{paramMG} = n_f + \frac{1}{2} n_f \left(n_f - 1\right)$$

and parameters for mixture of Gaussians is given by:

$$n_{paramMG} = n_{MG}\left(n_{paramG} + 1\right)$$

Both parameters for Gaussian and mixture of Gaussians can be calculated by maximizing the likelihood L(μ, ∑), which can be done by following equations:

$$\hat{u} = \frac{1}{|DS_T|} \sum_{x_i \in DS_T} x_i$$

$$\Sigma' = \frac{1}{|DS_T|} xi \sum_{x_i \in DS_T} (x_i, \hat{u})^\wedge T$$

The performance of Gaussian based models on metrics such computational and storage cost can be significantly less with a major disadvantage in that these models are sensitive to noise in training data. This disadvantage can induce bias and consequently decrease the accuracy of the model.

Other methodologies include Markov Model, Parzen Density Estimation, K-Nearest-Neighbors, which can be used depending on how robust / flexible a classifier is expected to behave and the kind of data being used to train the classifier.

2.1.6 Support Vector Data Descriptor (SVVD)

Defining a decision boundary is critical aspect of OCC. A dataset with positive only / positive only and minimal negative data points can easily misclassify objects and can be affected by noise in data. A SVVD aims to define hypersphere with a minimum volume (radius) to cover all the training dataset with minimizing the noise factor and minimizing mislabeling errors.

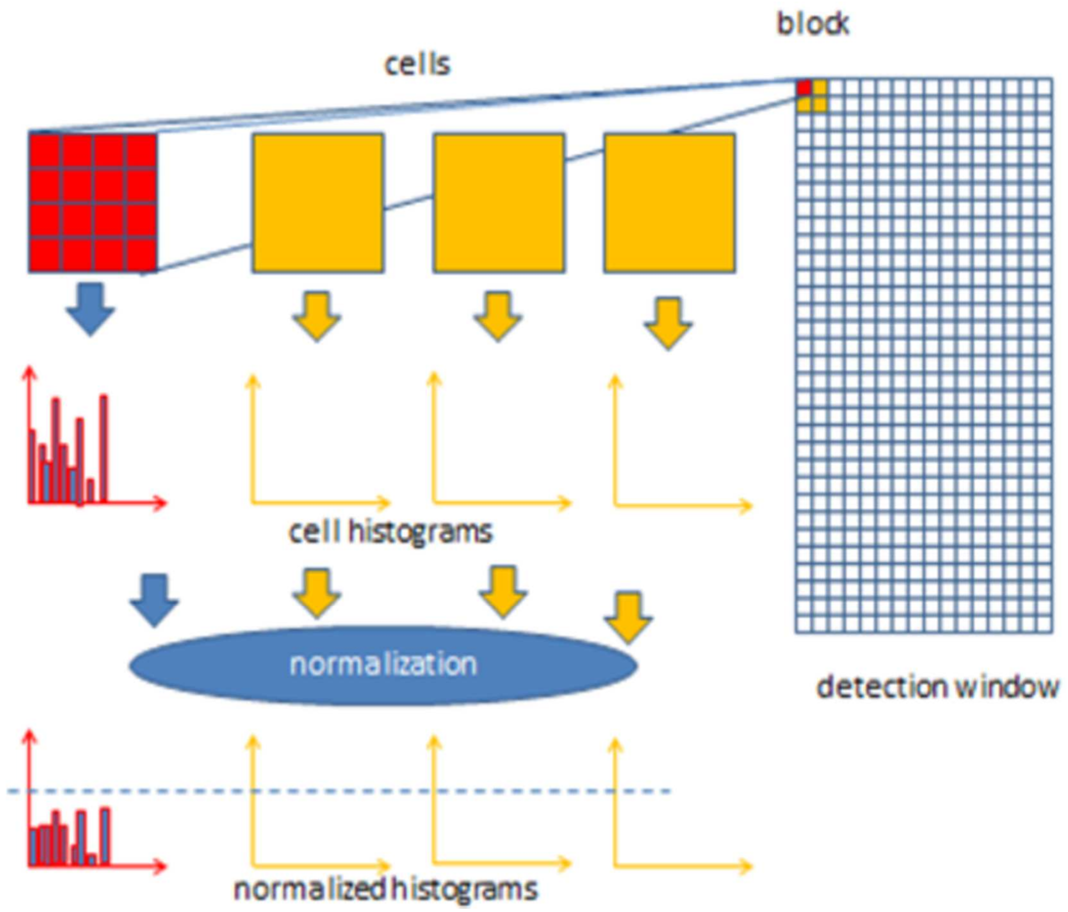


Figure 3: A visual representation of HoG Algorithm. Source: (Intel, n.d.)

The problem of rejecting data points lying outside the hypersphere can be seen. However, one way to include maximum points at greater distance is to use Polynomial or a Gaussian Kernel (Tax D. M., 2001). The use of kernel in the classifier can help in defining the hyperspherical boundary based on the how flexible or tight the boundary is desired and what kernel method is implemented.

Support vectors are the points on the boundary. Xi represents the outlier and has $\xi_i > 0$. The structural error as seen in above image can be defined as:

$$\xi_i(R, a) = R^2$$

to be minimized with following constraints:

$$||x_i - a||^2 \leq R^2, \forall i$$

Slack variables as introduced by (Tax D. M., 2001) to above equation to include all objects within the sphere and make the model more flexible. The equation proposed is as follows:

$$\varepsilon(R, a, \xi) = R^2 + C \sum_i \xi_i$$

along with constraints that almost all objects are within the sphere:

$$||xi - a||^2 \leq R^2 + \xi_i, \ \xi_i \geq 0, \quad \forall_i$$

SVVD proposed by (Tax D. M., 2001) is as following:

$$f_{SVVD}(z; \alpha, R) = I\left(||z - a||^2 R^2\right)$$

$$= \mathrm{I}\left((\mathrm{z.z}) - 2\sum_i \alpha_\mathrm{i}(\mathrm{z.x_i}) + \sum_{ij} \alpha_i \alpha_j (x_i, x_j) \leq \mathrm{R}^2 \right)$$

being an outlier object.

Inclusion of artificial outlier objects (Tax & Duin, 2001) has been proposed to optimize OSVM parameters and gain balance between over and under fitting of data points with respect to the sphere.

Defining decision boundary, as discussed earlier, is a critical aspect of OCC. In OSVM methodology many techniques have been proposed to improve the definition of classification boundary.

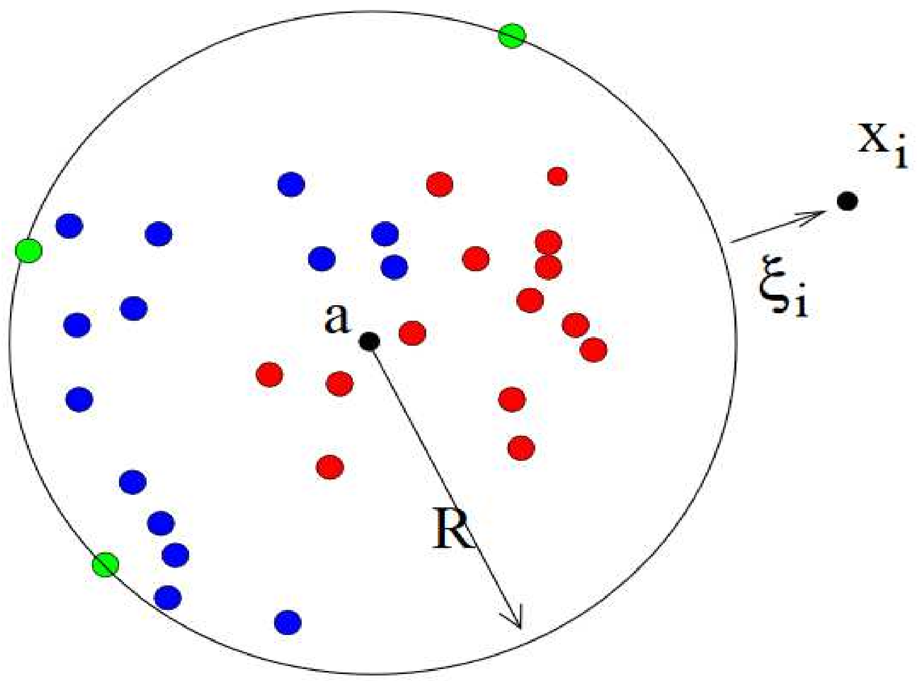


Figure 4: Source: (Tax D. M., 2001) A hypersphere containing target data with radius R and center at a.

Gaussian kernel is more effective than Polynomial and Linear kernels as argued by (Li, Huang, Tian, & Xu , 2003) to detect anomaly and define the outliers and propose improved version of OSVM by extending the work of (Schölkopf , Williamson , Smola , & Shawe-Taylor, 1999).

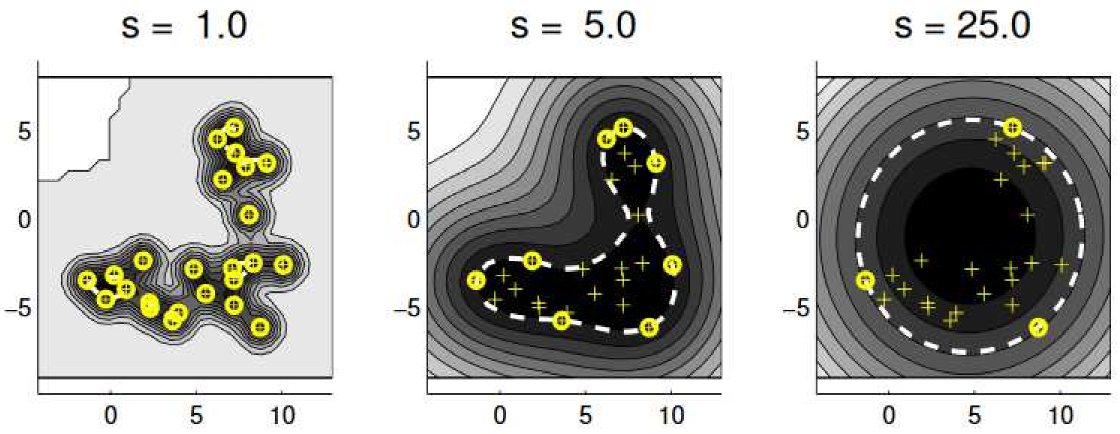


Figure 5: Source (Tax D. M., 2001). Data Description of banana-shaped dataset. The model uses a Gaussian kernel with varying s. Dashed line is description boundary and solid circle shows the support vector.

They consider origin to be a second class and all points “close enough” to origin as outliers.

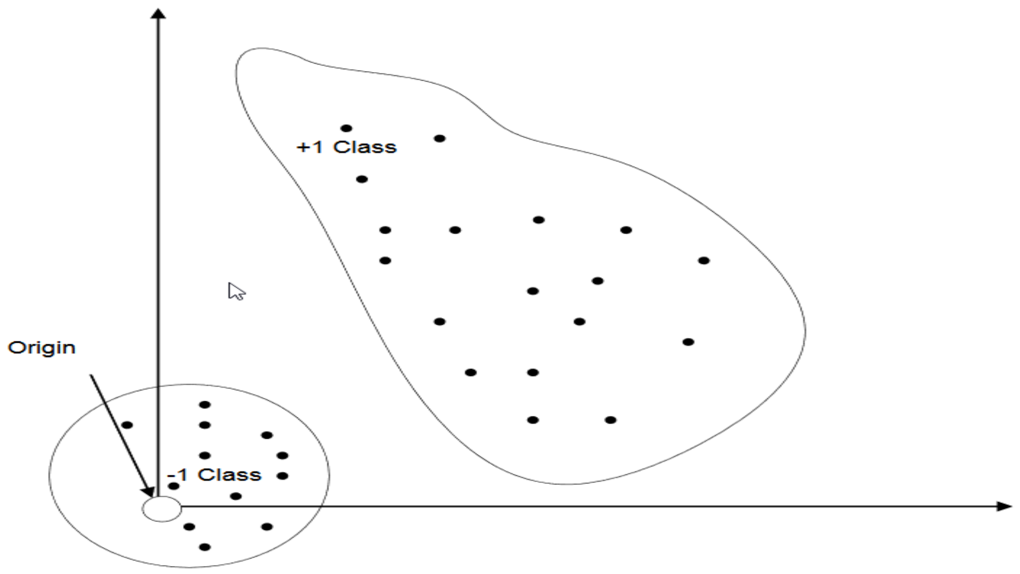


Figure 6: Source (Li, Huang, Tian, & Xu , 2003). An improved version of OSVM.

2.2 Active Learning

Active learning can be thought of an approach to skip large labeling of dataset. Active learning or “query learning” is a well-studied domain to overcome the labeling bottleneck. A survey by Settles (2010) on active learning and its methodologies explains scenarios, query strategy framework, analysis, problem setting variants, and practical considerations in active learning.

In context of using active learning for object detection in geo-spatial images, (Tuia, Volpi, Copa, & Munoz-Mari, 2011) present a survey of active learning algorithms for supervised remote sensing image classification. (Tuia, Volpi, Copa, & Munoz-Mari, 2011) present active learning algorithms for - committee-based, large margin-based, and posterior probability-based heuristics and compare performance of these algorithms. They conclude that large margin based methods with SVM model provide better accuracy than other models for hyperspectral images, however they argue that the best heuristic is problem-based with involves comparison of metrics such as accuracy, computational cost, and others.

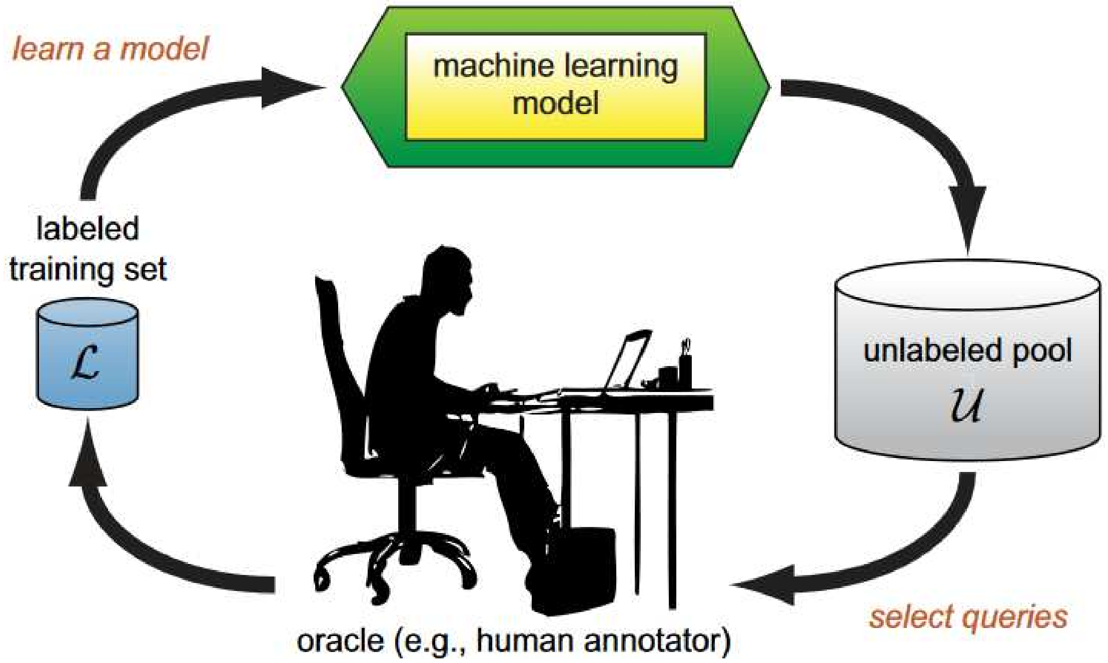


Figure 7: Source: (Settles, 2010). Pool Based Active Learning

Active learning methodology can also help to solve the problem of large labeling of dataset. In active learning, we first train a classifier, use the query based approaches to predict more labels, and then use an oracle (example- a human annotator) using query approach and then add the correct labels to the pool of data. This process eases out the manual labeling, however, the challenges such as high dimensionality of data, dependency on models for accuracy and oracle are persistent.

One way to address these problems is to combine robust classification models with strict boundary and feature reduction technique such as Principal Component Analysis (PCA). These steps will help in reducing effort for labeling data and build a robust classifier.
One such approach is proposed by (Vijayanarasimhan & Grauman, 2014). In their work, a part-based detector on top of a linear SVM and hash functions for querying the data has been proposed. The use of SIFT descriptor for feature encoding helps in building vectors to be efficiently used by the classifier and avoid mispredictions.
In the experiment chapter, we will discuss about the how active learning approach can be utilized to reduce the labeled dataset size requirements. A similar approach by (Yarowsky, 1995), where a small labeled dataset is used to train a classifier and is then used to label the unlabeled data. The accurate predictions are consequently added to the training set to build a more robust classifier. We in our experiment will extend this idea by using an incremental approach. We first train the classifier with a small labeled dataset along with random unlabeled samples and then measure the accuracy of the classifier. We will also test this idea by using a OCC to first choose samples from an unlabeled pool as outliers which are then added to a training set to check if this improves the accuracy of the overall model.

2.3 Support Vector Machines (SVM)

SVM is a widely-recognized technique for classification of binary class data introduced by (Cortes & Vapnik, 1995). The SVMs distinguishes two classes by constructing a hyperplane in large dimensional space. The points on the hyperplane are the support vectors which determine the decision boundary.

The approach proposed by (Cortes & Vapnik, 1995) for SVM is briefly summarized by (Meyer , 2017) in four parts namely- class separation, overlapping classes, nonlinearity, and problem solution. The class separation is the hyperplane separating two classes helps in defining the decision boundary and maximizing the distance between both classes. Overlapping classes or soft margin helps in reducing the effect of data points on “wrong” side of the margin which can lead to misclassification of a sample. Nonlinearity is when a linear plane cannot separate space and the data points are projected in higher-dimension. Problem solution is the formulation of the complete task which can be solved with known methodologies.

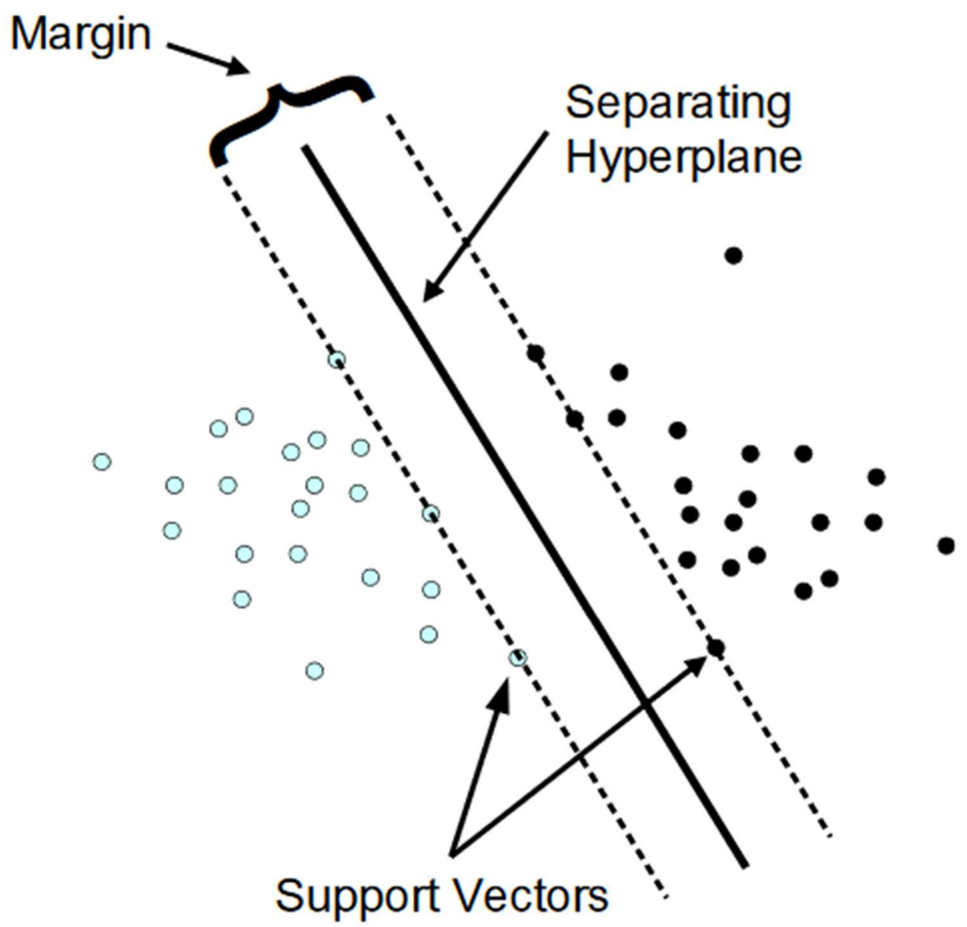


Figure 8: (Meyer , 2017). A plot showing two classes separated by hyperplane. The points on the boundary are support vectors.

The SVMs function better when used with kernel tricks. There have been many proposed kernel tricks such as Linear, polynomial, and quadratic kernels, to enable the classification with minimizing the errors and increasing the accuracy. The kernel tricks help in accurate decision boundary by clearly separating classes with both low and high dimensional feature space. This thesis work uses linear kernel trick in the experiments. Let us consider a sample set $S = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)$, where $x \in \mathcal{X}$ and labeled by $y_i \in \{-1, +1\}$. Let $S = (X, Y)$ be a labeled sample from set $\mathcal{X}$ that has empty intersection with hyperplane, then

$$\gamma = \min_{x \in \mathcal{X}} | \langle x, w \rangle + \theta | > 0$$

where w is the weight vector and θ is a bias of the hyperplane (Camphell & Cristianini). The γ being margin of the hyperplane w w.r.t. sample S. Also, the hyperplane w.r.t. sample is in canonical form:

$$\min_{x \in \mathcal{X}} | \langle x, w \rangle + \theta | = 1$$

further proving that,

$$\gamma = 1/\left|\left|w\right|\right|_2$$

In our experiments, we use SVM for binary classification and its comparison with OCC, which is an extended version of SVM for inlier/outlier detection. As discussed earlier, the SVMs are widely used and are known to provide high accuracy. This comparison of two methodology helps in determining if same or better accuracy can be achieved by using less labeled data.

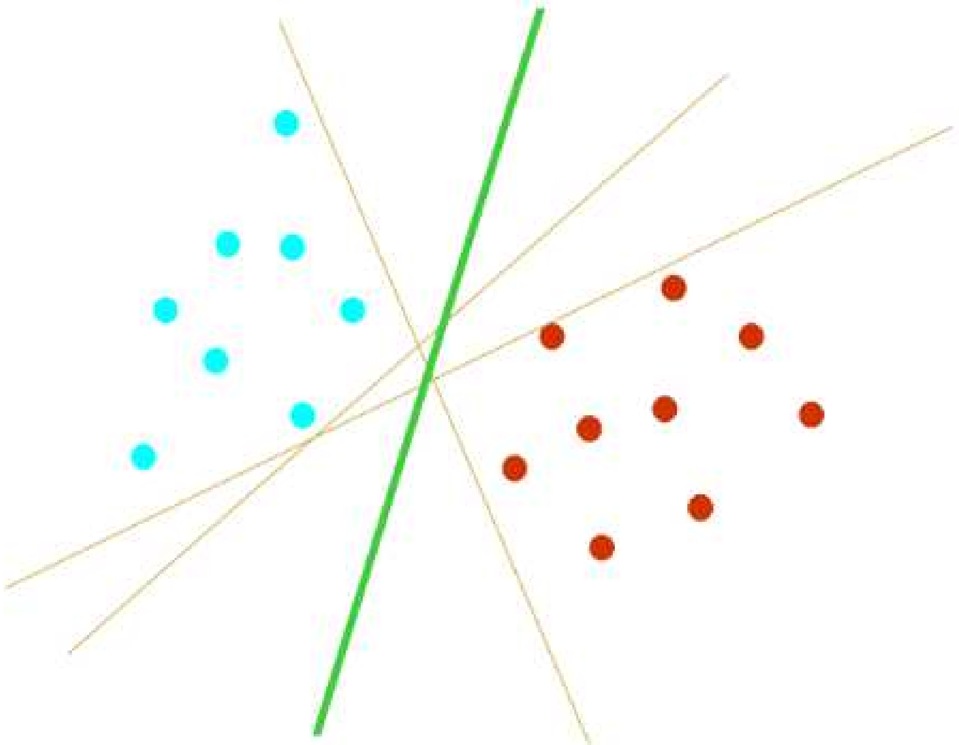


Figure 9: Source: (Gunn, 1998). Optimal Separating Hyperplane showing the separation between two classes. The figure also shows that the classifier generalizes well, which means, the unseen samples will be classified highly accurately.

2.3 Principal Component Analysis

Principal Component Analysis (PCA) is a technique which helps in lowering the feature space in high dimension data. PCA reduces data from n- dimensional feature space to k-dimensional feature space. The number of components in PCA is the dimensionality to which the data is reduced. To perform PCA on *n* dimensional space to reduce it to k dimensional feature space, we first compute the covariance matrix, which is given as follows:

$$\sum = \frac{1}{m}(x_i)(x_i)^T$$

and then compute eigenvectors of matrix $\sum$.

The linear transformation in PCA is the direction of maximum variance of data reduced to k-dimensional feature space which is mutually orthogonal.

The reduction of dimensional space helps in maximizing scatter of data points in linear projection. PCA can be defined as computation of principal components by performing eigenvalue decomposition of covariance of covariance matrix of complete training data (Chang, Nie, Yang, Zhang, & Huang, 2016). They define the objective function on PCA as:

$$\max_{W_T W=I} Tr(W^T X X^T W)$$

where Tr (.) denotes tracer operator.

CHAPTER 3

APPROACH

A prominent challenge when presented with large dataset is to first label the data so that we can classify the data into different classes. The labeling of the data is an expensive process because data with high dimensional feature space presents a lot of challenges as the features can be repetitive and they may or may not add valuable information about a sample. Curse of dimensionality is a difficult problem to handle especially when dataset is huge. Several datasets can also have missing values which are hard to compensate for and can lead to misclassification, resulting in unreliable accuracy of a classifier. The comparison of binary versus one class classification can help in understanding how much data is necessary in achieving high accuracy and in building a generic classifier to predict from unseen data.

The goal of this thesis is to lower the labeled dataset size requirement by comparing different classification methodologies for their accuracy and F1 score on different dataset sizes. We then extend the approach for active learning/semi-supervised learning to check if we can label the unlabeled data from a classifier trained on small dataset. We follow an incremental approach for varying dataset sizes and calculate how much data is required to improve the accuracy of a classifier. In the active learning experiment, the combination of two classification methodologies (OCC to choose samples from unlabeled pool and SVM to train on labeled data plus selected samples from OCC) tells if such an approach can be utilized to automate labeling of unseen data with high accuracy.

CHAPTER 4

EXPERIEMENTS

To compare the accuracy of the approach as explained in chapter 3, an SVM classifier is compared with a One Class SVM classifier trained only on positive samples. The accuracy and other parameters such as precision, recall, F1-score, and Support are compared with and without applying PCA to training data. The use of PCA is to check weather dimensionality reduction improves the accuracy on small datasets.

Accuracy is one of the metrics we use to compare performance of the classifiers. This thesis work does not focus on improving the accuracy, but evaluates the performance of a classifier without involving pre-processing techniques or using kernel tricks to help improve the accuracy and other metric score of the classifier. With rule based learning and its application with different techniques, Duch, Setiono, & Zurada (2004), in their work, achieve higher accuracy for same dataset as used in this thesis varying from +4 to +10 in percentage points.

4.1 Experimental Setup

The experiments were setup on a 64-bit Windows OS on top of Intel(R) Core (™) i7-6500 CPU @ 2.50 GHz 2.59 GHz processor with 16.0 GB of RAM.

The dataset used for the experiments was Pima Indian Diabetes database acquired from (Lichman, 2013). The database consists of a total of 768 instances of patients showing indications of being positive or negative for diabetes. Each data point has a total of 8 attributes with each labeled as 1 (for positive) and 0 (for negative) indication for diabetes.

4.2 Experimental Environment

Python 2.7 with scikit-learn (Pedregosa, et al., 2011) and imbalance-learn library (Lemaˆıtre, Nogueira, & Aridas, 2017) on PyCharm IDE.

Dataset Characteristics	**Number of Instances**	**Attribute Characteristics**	**Number of Attributes**	**Associated Task**
Multivariate	768 268 Positive 400 Negative	Integer, Real (all represented as real numbers in this experiment)	8	Classification

Table 2: Dataset Overview, Source (Lichman, 2013)

4.3 Experiment 1

In the first experiment a total of 268 samples were picked for both SVM and One Class SVM. The One Class SVM was first trained and tested for Positive only dataset and total instances of positive instances in the dataset is 268. For a fair comparison of both the models, SVM was also trained and tested on 268 samples with 102 positive and 166 negative samples. The positive and negative samples were randomly picked as the SVM's classification is not dependent on only positive samples.

Train and test split methodology for cross validation was applied to use 80% of 268 (214 samples) samples for training the classifier and 20% of 268 (54 samples) samples to test the classifier. Following are the results obtained for the first experiment showing the accuracy and precision score. Before diving into statistics of results, let us first define accuracy and precision.

Accuracy is a metric that provides correctly classified instances of a classifier and *precision score* is the measure of the ratio of accurately predicted samples over total predicted positive observations.

The accuracy achieved for 2-Class SVM in for experiment 1 is 68% and the precision score is 55%. The following table is a confusion matrix for 2-Class SVM results of prediction made on samples from the test data.

Confusion Matrix for SVM:

	Predicted Class 0	**Predicted Class 1**
Actual Class 0	True Negative (TN) = 27	False Positive (FP) = 8
Actual Class 1	False Negative (FN) = 9	True Positive (TP) = 10

Table3: Confusion Matrix for SVM classifier.

From the table above we see that the true negative is 27, true positive is 10, false negative is 9, and false positive is 8 and hence the accuracy (A), precision (P), recall (R), and F1-score is calculated as:

$$A = \frac{TP + TN}{Total} = \frac{10 + 27}{54} = 0.68$$

$$P = \frac{TP}{TP + FP} = \frac{10}{10 + 8} = 0.55$$

$$R = \frac{TP}{TP + FN} = \frac{10}{10 + 9} = 0.52$$

$$F\ measure = F_\beta = (1 + \beta^2) Precision \times \frac{Recall}{(\beta^2 Precision + Recall)} = 0.54$$

Precision, Recall, and F1-score for SVM classification:

	Precision	**Recall**	**F1-Score**
Positive for Diabetes	0.56	0.53	0.54

Table 4: Precision, Recall, F1-score (SVM)

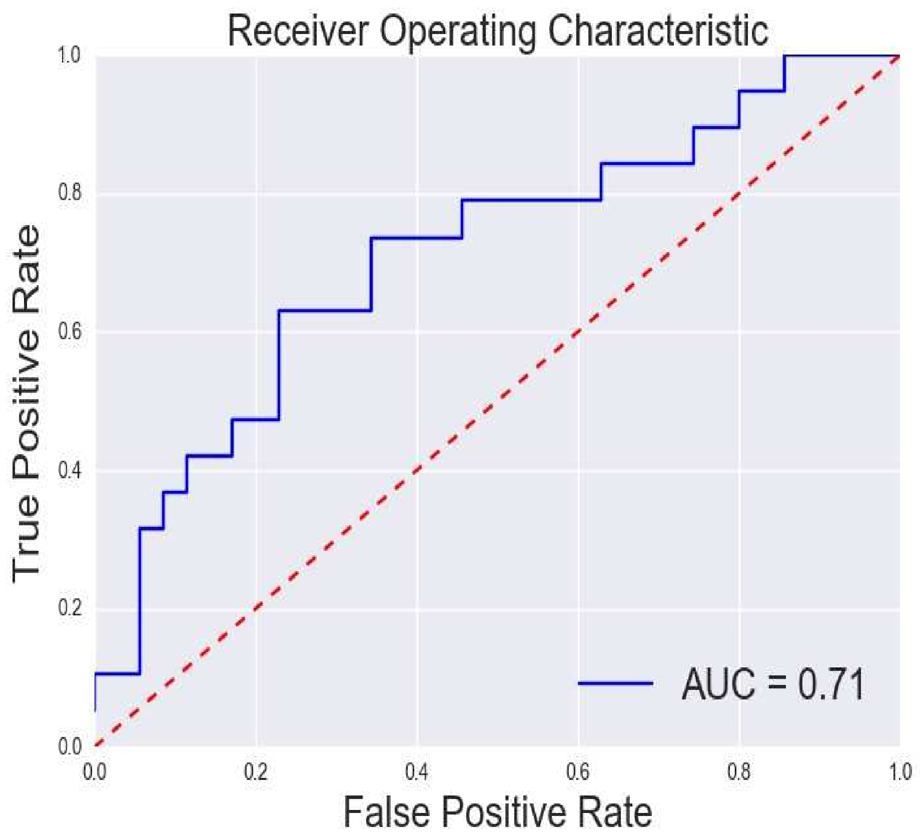


Figure 10: Graph describing ROC curve for this experiment.

From the results above we see that the classifier's holdout accuracy is at 68.51% and its precision score is 62.22% for predicting both negative and positive class.

Results for One Class SVM including the parameter *nu* from Python's Sklearn One Class SVM parameters:

nu is the parameter to tune the classifier for minimizing training errors. *nu* defines upper bound on fraction of margin errors and lower bound on support vectors which are relative to the total training samples.

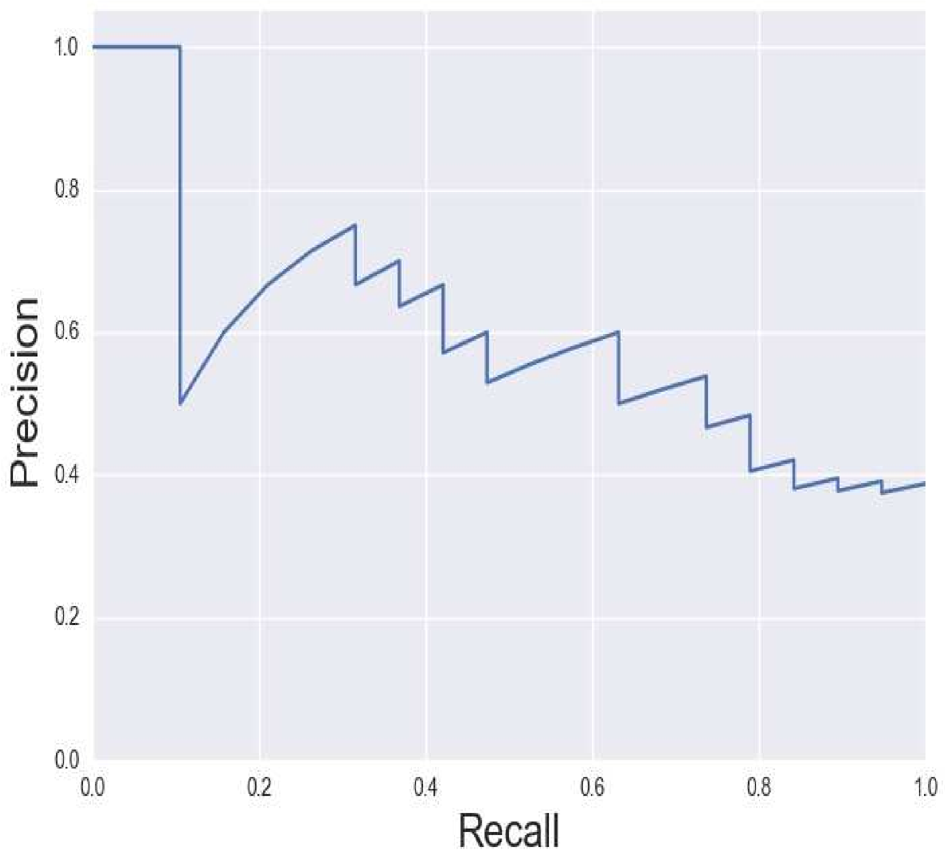


Figure 11: Precision-Recall Curve.

Following matrix shows how the accuracy increases when tuning nu parameter:

Nu	0.5	0.6	0.7	0.8	0.9
Accuracy	0.38	0.42	0.43	0.54	61

Table 5: Effect of *nu* Parameters on Accuracy for OCC.

The accuracy for OCC is at 61% when all the training data is positive samples only, and the precision score is 73.4%.

Confusion Matrix for One Class SVM:

	Predicted Negative	**Predicted Class 1**
Positive for Diabetes	TN = 12	FP = 16
Actual Class 1	FN = 5	TP = 21

Table 6: Confusion Matrix for OCC.

Precision, Recall, F1-score, Support for One Class SVM:

	Precision	**Recall**	**F1-Score**
Positive for Diabetes	0.57	0.81	0.67

Table 7: Precision, Recall, F1-score, and Total number samples.

One Class SVM in experiment 1 outperforms the 2-Class SVM. This can be measured by looking at the F1-score parameter for both the classifiers. The OCC was trained on positive samples only, while the 2-Class SVM was trained on both positive and negative samples. The decision for both classifiers is binary and as expected, the OCC would do better with small dataset as the decision boundary will include all maximum possible variance and improve the recall rate.

4.4 Experiment 2

In this experiment, we apply Principal Component Analysis (PCA) on our data set to check if reducing dimensionality to a lower dimensional feature space of dataset helps in improving the accuracy of a classifier. In this section, dataset size is 268 samples because of total 268 positive samples available.

Overall Results for SVM with PCA Dataset size = 268 with 102 positive and 166 negative samples:

The accuracy score in this experiment for 2-Class SVM is 69% and the precision score is at 60%.

Confusion Matrix for SVM with PCA:

	Predicted Negative	**Predicted Positive**
Positive for Diabetes	FN = 11	TP = 8

Table 8: Confusion Matrix for SVM with PCA

Precision, Recall, F1-score (SVM with PCA)

	Precision	**Recall**	**F1-Score**
Positive for Diabetes	0.57	0.42	0.48

Table 9: Precision, Recall, F1-score SVM with PCA

Results One Class SVM with PCA, dataset size = 268 total samples, 214 positive samples to train and 54 negative and positive samples to test:

The accuracy score achieved is 35% and the precision score is 56%.

Confusion Matrix One Class SVM with PCA:

	Predicted Negative	**Predicted Positive**
Positive for Diabetes	FN = 13	TP = 13

Table10: Confusion Matrix for SVM with PCA

Precision, Recall, F1-score (One Class SVM)

	Precision	**Recall**	**F1-Score**
Class 1	0.37	0.50	0.43

Table 11: Precision, Recall, F1-score One Class SVM

In this experiment, the 2-Class SVM outperforms the OCC with accuracy score of 69% to 35%. The PCA did not help in this scenario to improve the accuracy of OCC and F1 score. With low dimensionality of the feature space in this data set, reduction technique has not helped in projecting the linear data well for the decision boundary to include all the cases. Also, OCC trains on positive only samples which helps in building a strict decision boundary. With test samples from both negative and positive class, the confusion matrix for One Class SVM clearly shows that the precision and recall for OCC with PCA (with low feature space) is not the best methodology.

4.5 Experiment 3

For the active learning experiment, the entire dataset is divided into three namely Black (B) training data, Gray (G) unlabeled pool, and White (W) test data. Now, we first divide the dataset where 20% of data is named W for testing and is never utilized for training. The remaining 80% of the data is a pool to be utilized as B and G. Now from 80% available data, we first randomly sample 10% of data as B and treat 70% data as an unlabeled pool G.

4.5.1 The First Model

Algorithm 1 The First Model

```
1: $Dataset \leftarrow$ Total number of labeled samples
2: $W \leftarrow$ 20% of the Dataset
3: $B \leftarrow$ 10% of the Dataset-W
4: $G \leftarrow$ remaining 70% of Dataset
5: $RandomSamples \leftarrow$ 10% of G
6: for B + $RandomSamples \neq 80\%$ of Dataset do
7:     Train Binary SVM on B + Random Samples
8:     Test Classifier on W
9:     Capture F1-Score and Accuracy
10:    Random Samples = Random Samples + (G - Random Samples)*10%
11: Compare F1-Score and Accuracy for each iteration
```

Figure 12: Algorithm for The First Model

In the first model, we use binary SVM for classification and testing the model. We choose random samples from the unlabeled pool to select our random samples once the binary SVM is classified. Following is the procedure for the first model:

i. Train on 10% B data as chosen and randomly select 10% from G dataset.
ii. Increment 10% on every iteration and evaluate the holdout (W) data as the B data increases from 10% to 80%.
iii. Compare F1-score and accuracy for each iteration.

In this model, the approach is to check the F1 score and accuracy of the model when trained on 10% of the B data and 10% random samples from G data and measure the F1 and accuracy score based on the size of training data. All the iterations of classification this model is tested on same test data (White data) and the goal is to check if random samples are accurately labeled during classification.

The total positive samples in the dataset is around 35%, which is a classical undersampling problem. To balance the dataset, Random Over Sampler (python imbalance-learn) is used, which leads to the second model in for active learning methodology.

The results of the first model are shown in fig. 13. The figure shows a plot for Number of Samples v/s F1-score at each iteration of the increasing dataset size. The F1-Score reaches a plateau when the dataset size is around 65% of $B + G$ dataset.

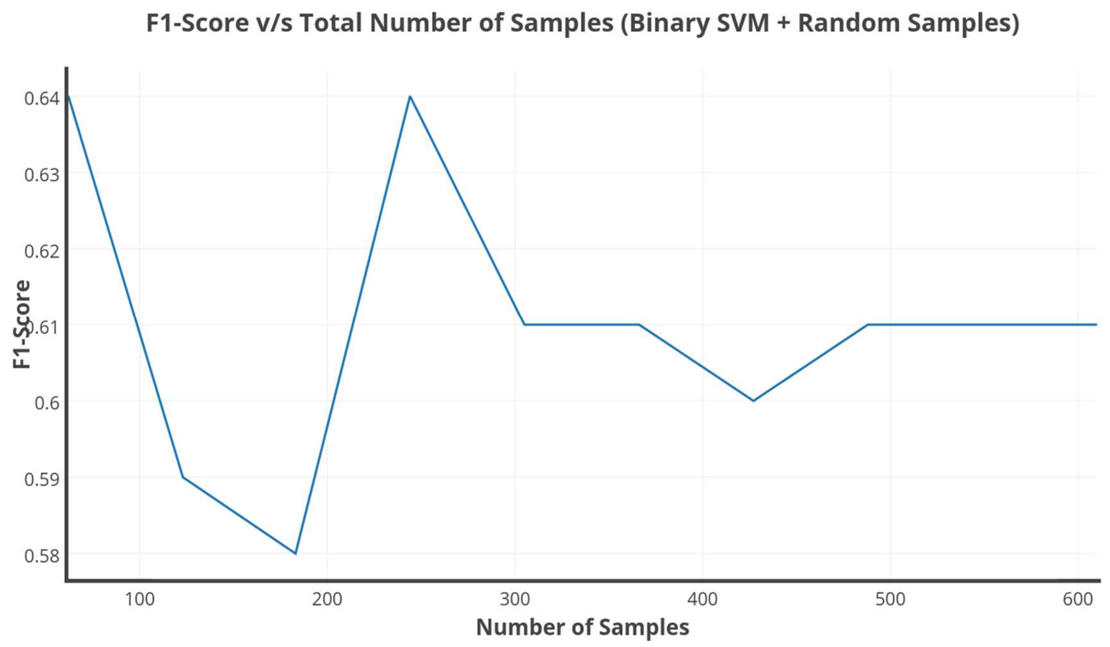


Figure: 13. The Graph Shows Number of Samples v/s F1-score For Each Iteration of The First Model.

4.5.2 The Second Model

In this model, we repeat the experiment as described in first model. Randomly choosing samples from B and G data can result in imbalance of data while training. To overcome such a scenario, we use Random Over Sampling methodology to resample with replacement (Lemaˆıtre, Nogueira, & Aridas, 2017) on B data for each iteration to keep the class frequencies balanced.

Algorithm 1 The Second Model

```
1: $Dataset \leftarrow$ Total number of $labeled\ samples$
2: $W \leftarrow$ 20% of the Dataset
3: $B \leftarrow$ 10% of the Dataset-W
4: $G \leftarrow$ remaining 70% of Dataset
5: $RandomSamples \leftarrow$ 10% of G
6: for B $+$ $RandomSamples \neq 80\%$ $of\, Dataset$ do
7:     Fit to ROS B + Random Samples    ▷ ROS to balance the data before training
8:     Train Binary SVM on B + Random Samples
9:     Test Classifier on W
10:    Capture F1-Score and Accuracy
11:    Random Samples = Random Samples + (G - Random Samples)*10%
12: Compare F1-Score and Accuracy for each iteration
```

Figure 14: Algorithm for The Second Model.

To measure the performance with Random Over Sampling, we calculate F1 score and accuracy of the classifier on each iteration and compare the scores with each increment.

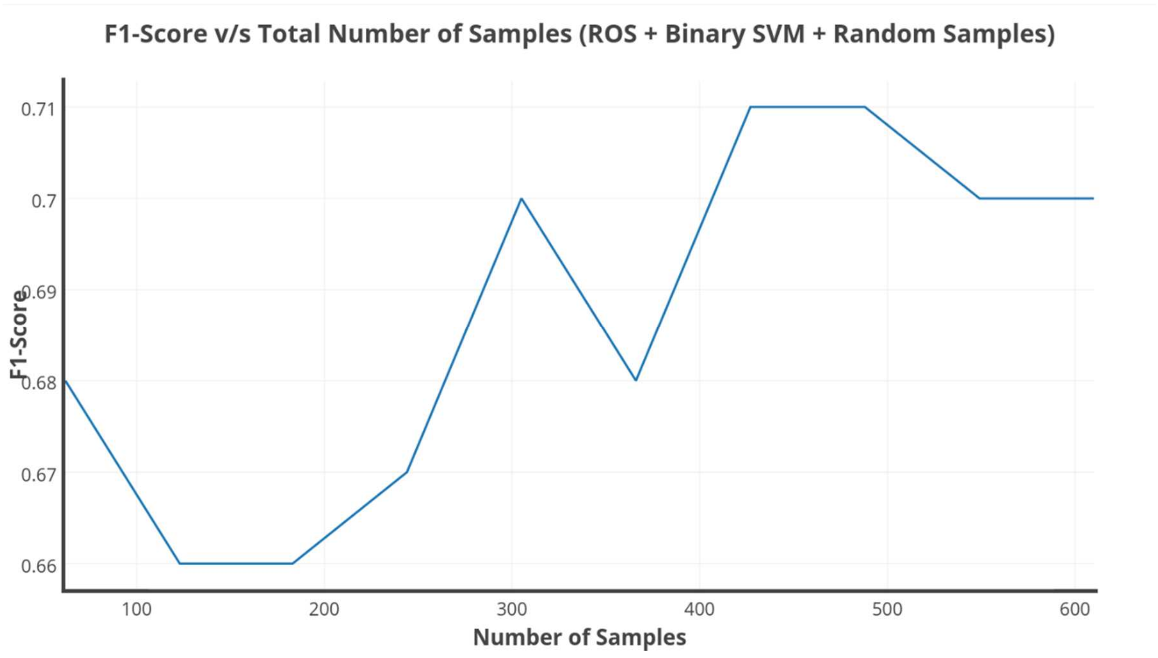
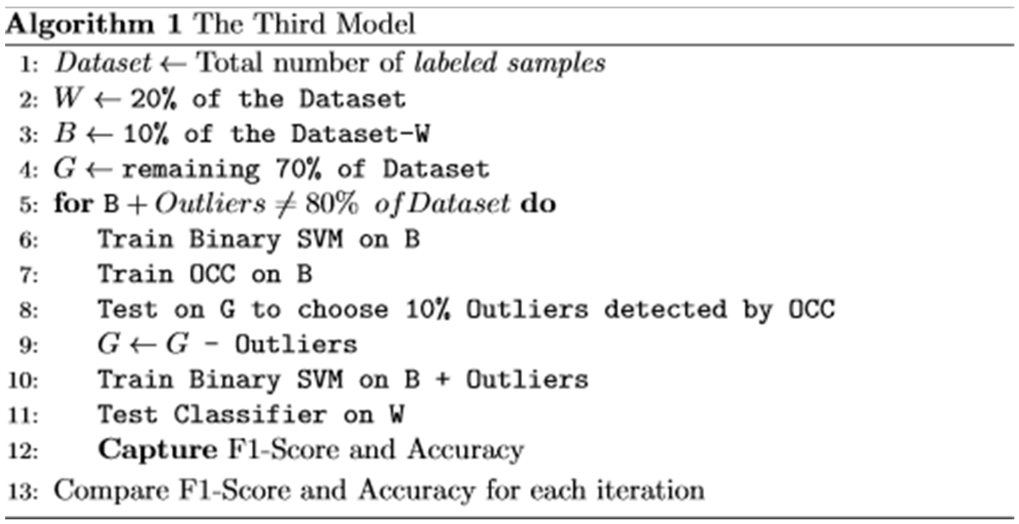


Figure: 15. The Graph Shows Number of Samples v/s F1-score For Each Iteration of The Second Model with Random Over Sampling.

The results of this experiment are shown in fig. 15. The graph plot of F1-Score v/s Number of Samples with Random Over Sampling shows that the F1-Score is higher than as observed in the first model, which concludes that the random over sampling technique has helped in improving both the accuracy and F1-score of this methodology.

4.5.3 The Third model



```
Algorithm 1 The Third Model
1: $Dataset \leftarrow$ Total number of $labeled\ samples$
2: $W \leftarrow$ 20% of the Dataset
3: $B \leftarrow$ 10% of the Dataset-W
4: $G \leftarrow$ remaining 70% of Dataset
5: for B + $Outliers \neq 80\%\ of Dataset$ do
6:     Train Binary SVM on B
7:     Train OCC on B
8:     Test on G to choose 10% Outliers detected by OCC
9:     $G \leftarrow G$ - Outliers
10:    Train Binary SVM on B + Outliers
11:    Test Classifier on W
12:    Capture F1-Score and Accuracy
13: Compare F1-Score and Accuracy for each iteration
```

Figure 16: Algorithm for The Third Model

The third model uses a combination of both binary-SVM and OCC classifiers. Following is the procedure for the third model:

i. Train a OCC with B as chosen from the dataset.

ii. Test OCC on G and pick 10% of outliers. These outliers are then subtracted from G

iii. The samples (outliers) chosen from OCC are then added to the B samples, which are used for training binary-SVM.

iv. The trained classifier is tested on W data for each iteration.

v. Compare F1-score and accuracy of the model for each iteration.

Picking outliers, as mentioned in step 2, also depends on the value of *nu* set during training the OCC classifier. Higher the *nu* value, stricter the classification boundary becomes. The *nu* value minimizes training errors and defines upper bound on fraction of margin errors and lower bound on support vectors which are relative to the total training samples.

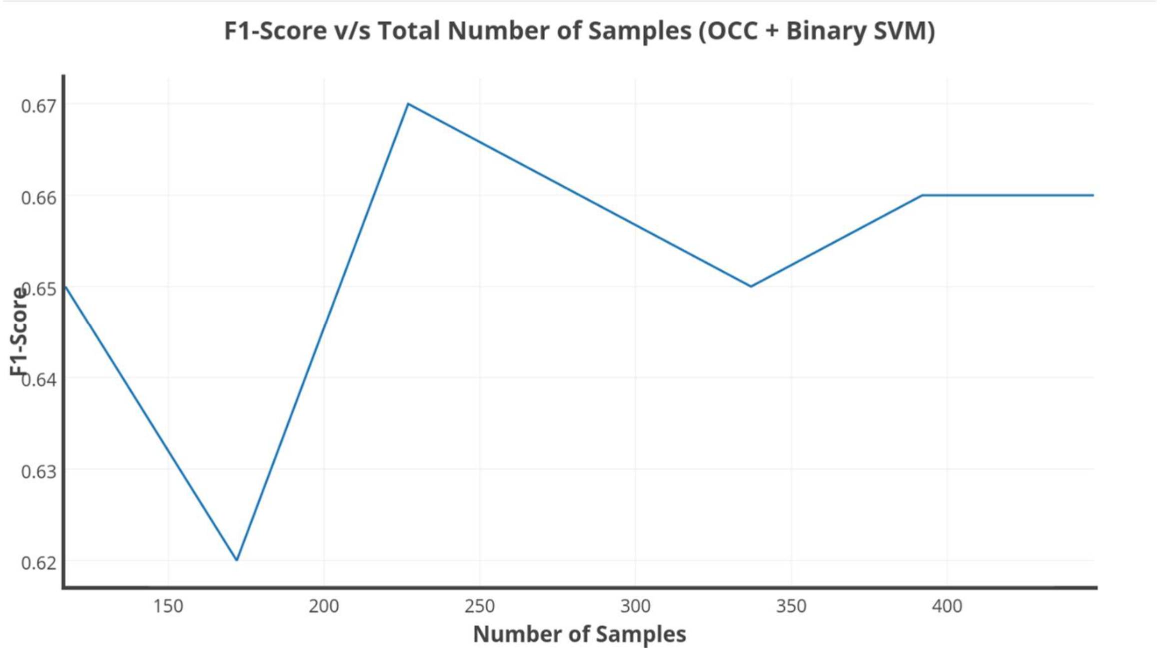


Figure 17: F1-Score v/s Number of Samples when using OCC + Binary SVM.

The results of this experiment are shown in fig. 17. In this experiment, we see that the OCC + Binary-SVM combination works better than first and second models. The F1-Score as see above is obtained when outliers are identified and added to Binary-SVM for classification and tested on W. The performance of third model is better in terms of both performance and dataset size. The total data used to train Binary-SVM in this model is smaller because we only add outliers to B knowing that the true positive samples as predicted by OCC have been correctly identified.

4.5.4 The Fourth Model

As discussed in the second model, the third model could use samples which are imbalanced. In the fourth model too, we use Random Over Sampling (ROS) methodology to rectify the imbalance in samples while training (Lemaˆıtre, Nogueira, & Aridas, 2017). The ROS methodology is applied to the samples before training the classifier. For each iteration, the comparison metrics F1-score and accuracy is calculated and comparison is drawn for every iteration.

```
Algorithm 1 The Fourth Model
 1: Dataset ← Total number of labeled samples
 2: W ← 20% of the Dataset
 3: B ← 10% of the Dataset-W
 4: G ← remaining 70% of Dataset
 5: for B + Outliers ≠ 80% of Dataset do
 6:     Fit to ROS B              ▷ ROS to balance the data before training
 7:     Train Binary SVM on B
 8:     Fit to ROS B              ▷ ROS to balance the data before training
 9:     Train OCC on B
10:     Test on G to choose 10% Outliers detected by OCC
11:     Fit to ROS B + Outliers   ▷ ROS to balance the data before training
12:     G ← G - Outliers
13:     Train Binary SVM on B + Outliers
14:     Test Classifier on W
15:     Capture F1-Score and Accuracy
16: Compare F1-Score and Accuracy for each iteration
```

Figure 18: Algorithm for The Fourth Model

The results of this experiment are shown in fig. 19. In this experiment, we see that Random Over Sampling technique used to balance the data for OCC and Binary-SVM helps in increasing the performance of the proposed model.

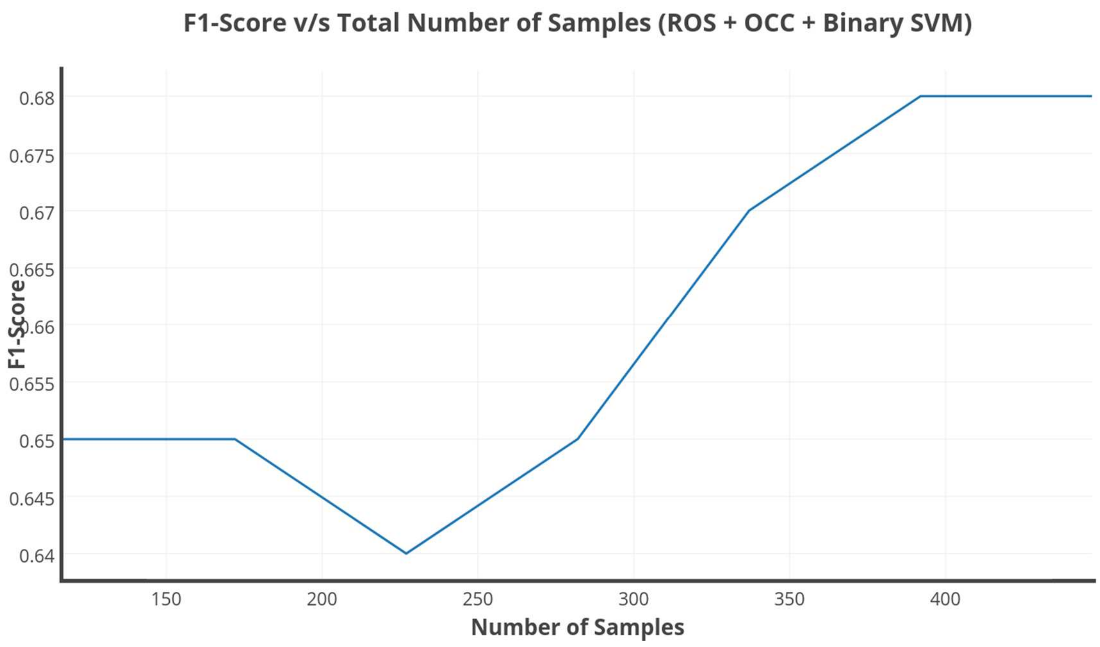


Figure 19: F1-Score v/s Number of Samples for ROS + OCC + Binary SVM.

The following is a combined result of four models as discusses:

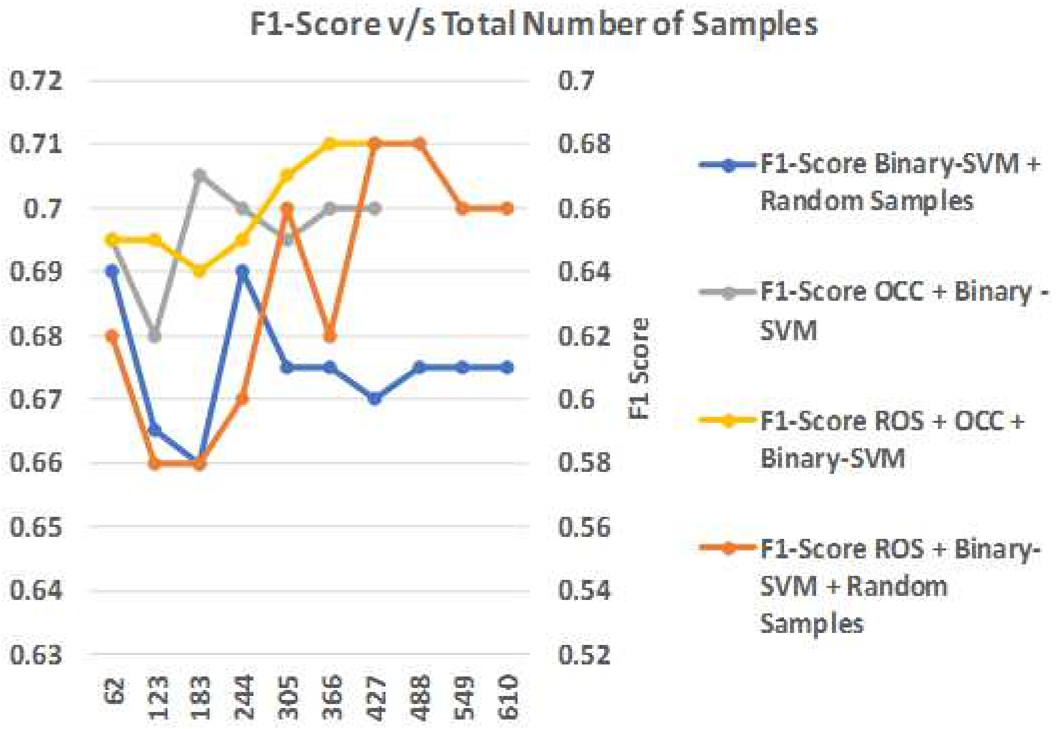


Figure 20: Combined Results for Experiment 3 Diabetes Data.

4.6 Repeating Experiments for a larger dataset

To test the proposed idea, we repeat experiments for the dataset as shown in table 12 acquired from UCI machine learning repository.

Dataset Characteristics	**Number of Instances**	**Attribute Characteristics**	**Number of Attributes**	**Associated Task**
Multivariate	30,000 6,336 Positive 23,664 Negative	Integer, Real (all represented as real numbers in this experiment)	24	Classification

Table 12: Dataset Overview, Source (Lichman, 2013)

The experiments were setup same as in section 4.4 and the dataset samples are show which clients are likely to Default on Credit Card (DCC). Let us know look the results obtained when the dataset size increases from 768 to 30,000 samples.

4.6.1 Results of Experiment 1 repeated for DCC dataset

	Accuracy	**F1 Score**
OCC	79%	0.82
SVM	82%	0.80

Table 13: Results of OCC v/s Binary SVM for DCC dataset.

From table 13 we see that the OCC performs better than Binary-SVM. The dataset does not use any data pre-processing techniques the classifier is trained on 80% of data and tested on 20% of the data. The results have been obtained after repeating the experiment five times and verifying that the results do not fluctuate more than 0.20 standard deviation.

4.6.2 Results of Experiment 2 repeated for DCC dataset

	Accuracy	**F1 Score**
OCC	65%	0.64
SVM	81%	0.75

Table 14: Results of OCC v/s SVM with PCA on DCC dataset

From table 13 we see that when the dimensionality of the dataset is reduced Binary-SVM performs better than OCC. The reduction in dimensionality (PCA) on training data for OCC might lead to loss of important information especially when the dataset only consists of positive samples versus both positive and negative samples as in case of Binary-SVM.

4.6.3 Results of Experiment 3 repeated for DCC dataset.

4.6.3.1 Results for The First model.

The results of this experiment are shown in fig. 21. With larger dataset size, we see that the F1-Score as compared to the results in section 4.4.1 is higher. The F1-Score drops in second iteration of loop however, it increases as the dataset size grows larger. A plateau can be seen in the last two iterations of the experiment when the dataset size reaches 80% of the training data.

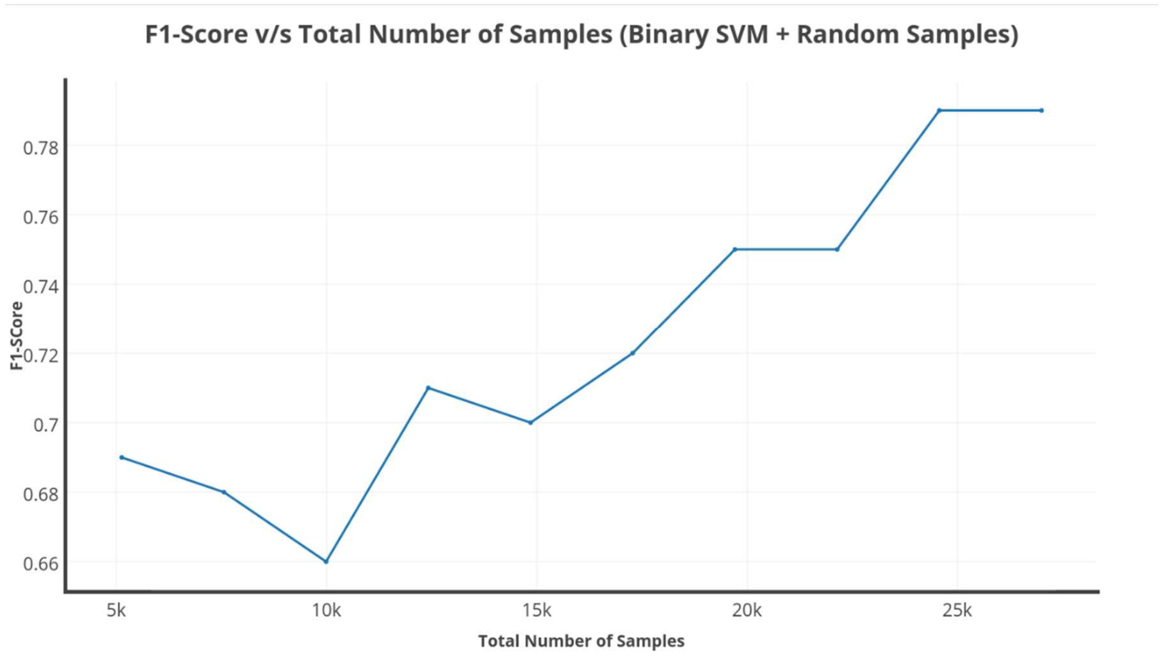


Figure 21: Number of Samples v/s F1-Score for Binary SVM + Random Samples.

4.6.3.2 Results for The Second model

The results of the second model with DCC show that with Random Over Sampling the performance of the classifier increases. The F1-Score compared to previous experiment is higher and consistently increases with increase in the dataset size.

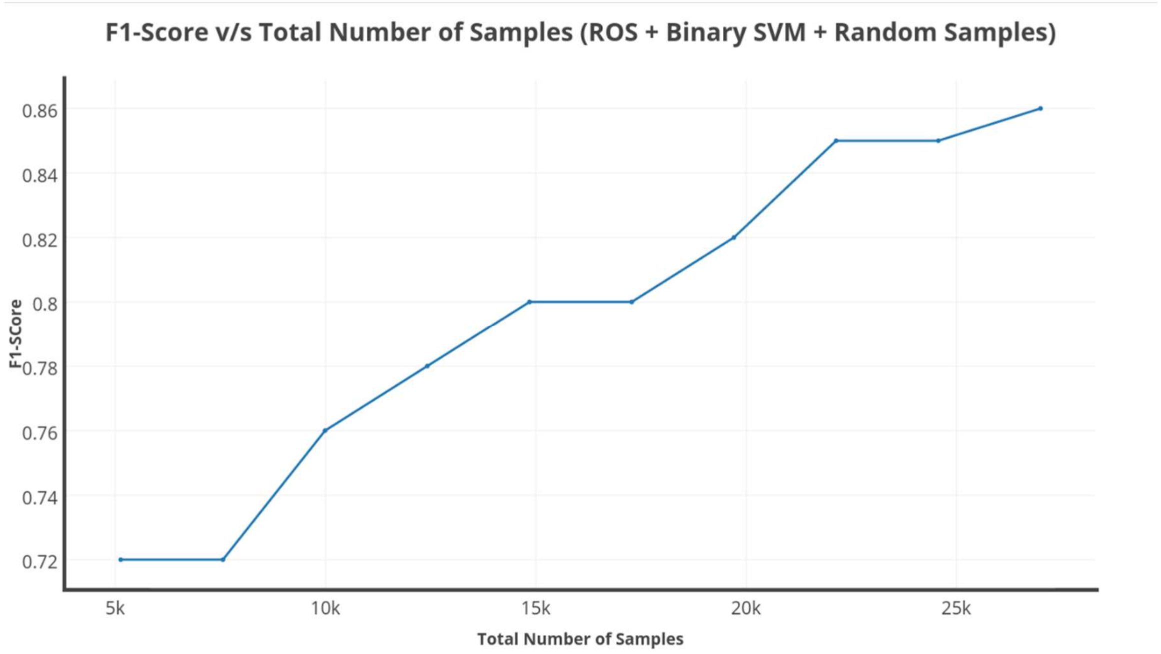


Figure 22: Number of Samples v/s F1-Score for ROS + Binary SVM + Random Samples

4.6.3.3 Results for The Third model

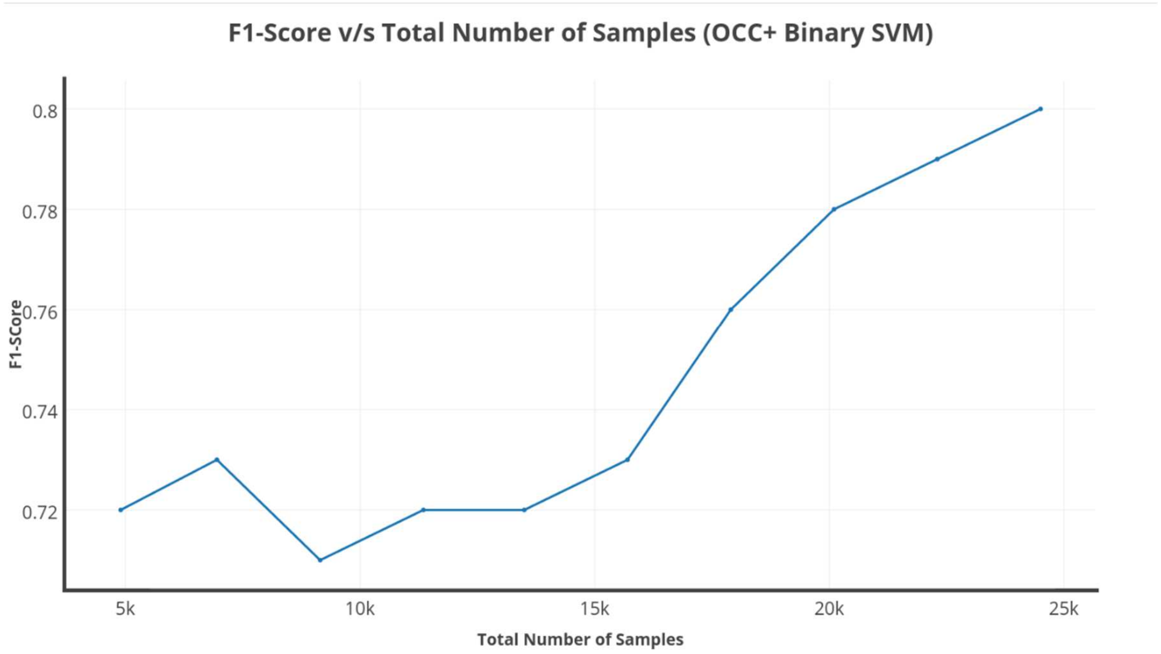


Figure 23: Number of Samples v/s F1-Score for OCC + Binary SVM

In fig. 23 we see the results for the third model. Here we repeat the experiment as in section 4.3.3 in which, we use a combination of OCC and binary-SVM. With larger dataset size to train from, the F1-Score is better than observed in section 4.4.3. The F1-Scores increases with every iteration of the loop when expect for the third iteration. Also, we see that the total number of samples to train binary-SVM is less than total number of samples used in experiment 1 and 2. This is so because, we first train the OCC to detect and choose outliers from B data and not add correctly labeled samples when training Binary-SVM.

4.6.3.4 Results for The Fourth model.

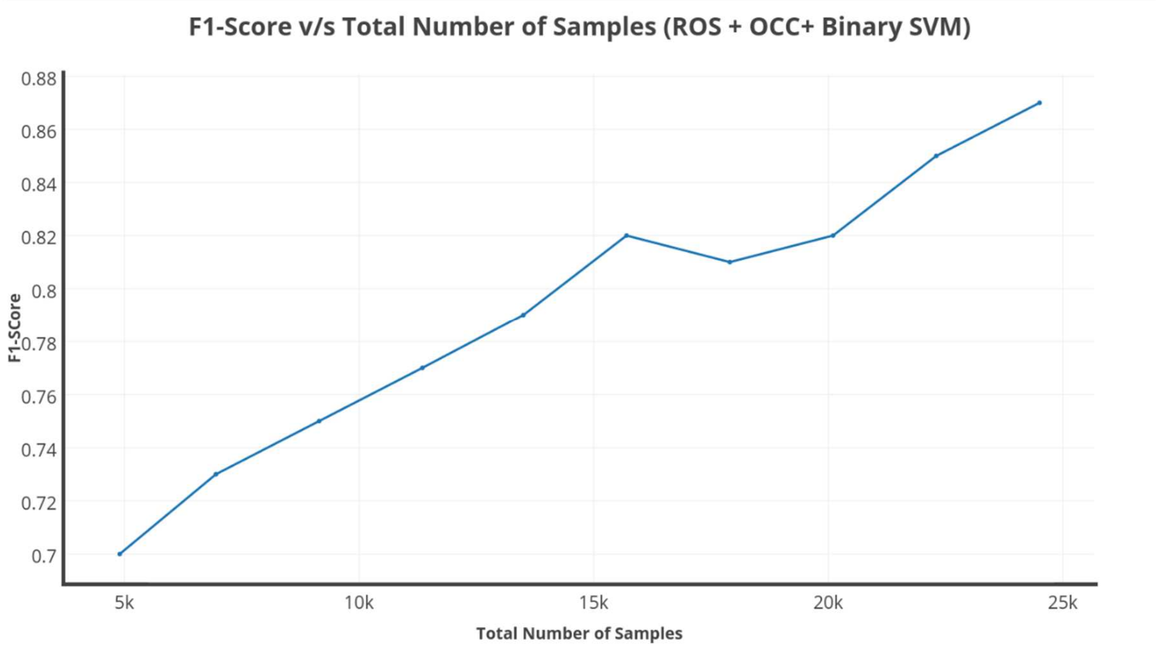


Figure 24: Number of Samples v/s F1-Score for ROS + OCC + Binary SVM

The results for the fourth model is shown in fig. 24. Here we use Random Over Sampling before training OCC and binary SVM, keeping the rest of the methodology same as in the third model. Here we see that, the F1-score is higher than as observed in the third model and outperforming the first and second model.

Now let us look at the combined results for experiment 3 for DCC dataset:

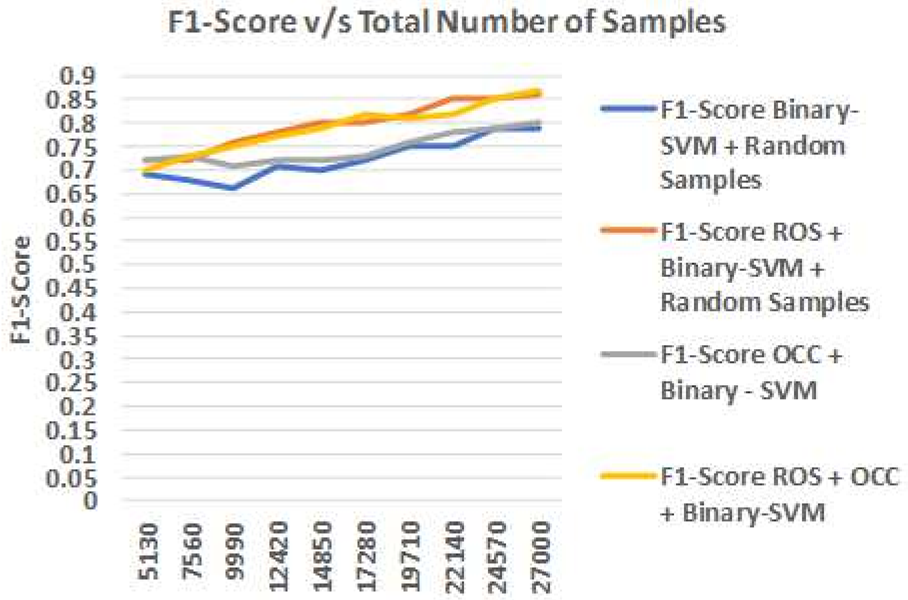


Figure 25: Combined Results of Experiment 3 for DCC Dataset.

CHAPTER 5

CONCLUSION

The work presented in this thesis shows that with small dataset, accurate predictions can be made. The accuracy of the model can be increased with dimensionality reduction techniques. The comparison of OCC and binary SVM tells that for small dataset with less features, OCC performs better than binary SVM, however with feature reduction, binary SVM outperforms the OCC. The concept of using less data to achieve high accuracy can help in reducing costs and effort for labeling the data. Furthermore, we can also build active machine learning models with less data and automate labeling of data with combination of classifiers.

The hypothesis holds true in experiment 1, where OCC outperforms the binary SVM, however, with increasing dataset size and inclusion of outliers, the OCC does not tend to do better in experiment 2. One of the reasons for this could be the imabalance in data which is seen for both Diabetes and DCC dataset. In the Experiment 3we see that Binary SVM performs better when used with Random Over Sampling technique, however, OCC + Binary-SVM with and without Random Over Sampling performs better. In the third and fourth model as proposed in experiment 3 we see that the performance is better both in terms of F-1 score and accuracy, and that the dataset size used is smaller for the third and fourth model. On repeating the experiments with on DCC dataset, we see that our methodology provides expected results as seen in Diabetes dataset. OCC has performed steadily in experiment 3, which tells that combination of two classifiers is a competitive approach to label the unseen data, hence reducing the requirement of large dataset.

We also see in our experiments that the data-preprocessing is valuable. Use of random over sampling improves the accuracy for all the models, however, we in this thesis focus on reducing the dataset size requirements. One of the other pre-processing techniques to overcome imbalance in data is to assign weights to the samples before training the classifier.

CHAPTER 6

FUTURE WORK

The approach as presented in this work can be extended to image classification for object detection. Quantum GIS is a very popular tool used for labeling objects in geo-spatial images, however, it requires manual effort for labeling both positive and negative samples. With abundant unlabeled geo-spatial imagery, the active learning approach as presented can be utilized to label the data beginning with using small data set of labeled images. The work of (Dalal & Triggs, 2005) shows an approach to detect humans using active learning approach. Another approach could be- using QGIS, label objects in geo-spatial image, extract all the labeled image patches disintegrating large geo-spatial images into small image samples and mark them as positive or negative sample. A combination of, OCC to detect outliers and then using a SVM to classify images, will make a robust model to label the large unlabeled pool of images. Also, images can have very large dimensional space. Using feature reduction techniques such as PCA, HaaR, Scale Invariant Feature Transform(SIFT), and HoG can help in reducing large dimensionality and build a more robust feature space, consequently boosting the classifier's accuracy.

REFERENCES

Barnab-Lortie, V., Bellinger, C., & Japkowicz, N. (2015). Active Learning for One-Class Classification. *IEEE 14th International Conference on Machine Learnng and Applications.*

Camphell, C., & Cristianini, N. (n.d.). *Simple Leaning Algorithms for Training Support Vector Machines.*

Chang, X., Nie, F., Yang, Y., Zhang, C., & Huang, H. (2016). Convex Sparse PCA for Unsupervised Feature Learning. *ACM Transactions on Knowledge Discovery from Data (TKDD)*.

Cortes, C., & Vapnik, V. (1995). Support-vector network. *Machine Learning*, 1-25.

Dalal, N., & Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection . *CVPR 2005 : proceedings : 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition.* Los Alamitos, Calif. : IEEE Computer Society 2005.

Duch, W., Setiono, R., & Zurada, J. M. (2004). Computational intelligence methods for rule-based data understanding . *Proceedings of the IEEE* (pp. 771-805). IEEE.

Gunn, S. R. (1998). *Support Vector Machines for Classification and Regression.*

Intel. (n.d.). *Histogram of Oriented Gradients (HOG) Descriptor* . Retrieved from Intel: https://software.intel.com/en-us/node/529070

Khan, S. S., & Madden, M. G. (2004). One-Class Classification: Taxonomy of Study and Review of. Cambridge University Press.

Lemaˆıtre, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A Python Toolbox to Tackle the Curse of. *Journal of Machine Learning Research*, 1-5.

Li, K.-L., Huang, H.-K., Tian, S.-F., & Xu , W. (2003). Improving one-class svm for anomaly detection. *In Proc. of International Conference on Machine Learning and Cybernetics.*

Lichman, M. (2013). *"{UCI} Machine Learning Repository*. Retrieved from University of California, Irvine, School of Information and Computer Sciences: http://archive.ics.uci.edu/ml

Mazhelis, O. (n.d.). One-Class Classifiers: A Review and Analysis of Suitability., (pp. 29-48).

Meyer , D. (2017). Support Vector Machines.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel , A., Thirion, B., Grisel, O., . . . Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 2825-2830.

Schölkopf , B., Williamson , C. R., Smola , A., & Shawe-Taylor, J. (1999). SV Estimation of a Distribution's Support.

Settles, B. (2010). *Active Learning Literature Survey.*

Tax, D. M. (2001). *One-class classification Concept-learning in the absence of counter-examples.*

Tax, M. D., & Duin, P. R. (2001). Combining One-Class Classifiers. . *Multiple Classifier Systems.*

Tuia, D., Volpi, M., Copa, L., & Munoz-Mari, J. (2011). A Survey of Active Learning Algorithms for Supervised Remote Sensing Image Classification. *IEEE Journal of Selected Topics in Signal Processing* , 606-617.

Vijayanarasimhan, S., & Grauman, K. (2014). Large-Scale Live Active Learning: Training Object Detectors with Crawled Data and Crowds. *International Journal of Computer Vision*.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. *Proceedings of the Association for Computational Linguistics (ACL)* (pp. 189–196). ACL Press.