An Efficient Approach Parallel Support Vector Machine for Classification of Diabetes Dataset

Naveeen Kumar Shrivastava[1], Praneet Saurabh[2] and Bhupendra Verma[3]
Department of Computer Science & Engg,
Technocrats Institute of Technology,
Bhopal, India

ABSTRACT
The paper proposes a Parallel SVM for predicting the diabetes chances in human based on a survey dataset which relates the different body parameters with diabetic and non diabetic persons. The aim of the paper is to correctly predict the future possibility of diabetes for any person. Since the survey dataset size could be very large with large numbers of parameters which makes it difficult to handle by simple SVM hence a parallel SVM concept is proposed in this paper to distribute these datasets into n different sets for n different machines which reduces the computational complexity, processing power and memory requirements for each machine. The proposed method is simple but quite reliable for parallel operation of SVM and can be used for large and unbalanced datasets the method also provide the flexibility to modify according to the dataset size, processors and memory available on different units. We have tested the proposed method using MATLAB and results are very encouraging.

Keywords: Diabetes, Support vector Machine, K means Clustering, Parallel Support Vector Machine, Binary Classification.

1. INTRODUCTION
Diabetes is a chronic disease and a major public health challenge worldwide. According to IDF(international diabetic federation) there are currently 246 million diabetic people worldwide and this no is expected to rise to 380 million by 2025. 3.8milion deaths are attributing to diabetes complications each year. 80% of type 2 diabetes complications can be prevented or delayed by early identification of people at risk. Disease prediction can be used for many government organizations to manage their functionality and preparing for future risks, and for private sectors like insurance companies can make evidence based decisions and can optimize, validate and refine the rules that govern their business. There are several machine learning approaches. Lot of research carried out on this context in support vector machine. There are many methods are available for prediction but because natural process of this kind are very complex which involves large number of input variables so we need very large dataset for proper prediction. Present days the support vector machines are emerges as excellent classifier and also SVM has advantages over other techniques like artificial neural networks that solution from SVM is global and unique while ANN suffers from local minima other advantages are 1.) The SVM has a clear geometrical explanation and representation and the 2.) Complexity of SVM does not depend upon the dimensions of input data. 3.) The SVM utilizes the structural risk minimization not the empirical risk minimization as in case of ANN. 4.) SVM is less prone to over-fitting. The reasons of selecting is clear but it also has disadvantage from a practical point of view and this is the most serious problem with SVMs is the high algorithmic complexity and extensive memory requirements of the required quadratic programming in large-scale tasks [1]. Because this paper mainly deals with parallel processing of SVM now we are going to discuss the need of parallel processing. As we know that it's a machine learning technique and like many machine learning techniques, SVMs involve a training stage, where the machine learns a pattern in the data from training data set. The process of learning allows parameters to be adjusted towards optimal values, while guarding against over fitting. The training stage for Support Vector Machines involves at its core a dense convex quadratic optimization problem (QP). Solving this optimization problem is computationally expensive, primarily due to the dense Hessian matrix. Solving the QP with a general-purpose QP solver would result in the time taken to scale cubically with the number of data points $(O(n^3))$. Such a complexity result means that, in practice, the SVM training problem cannot be solved by general purpose optimization solvers [2]. The complexity cubically increases with data points hence for very large data sets its better to divide data points in parts which not only decrease the complexity but also provide the capability of handling the tasks in parallel on separate systems. Although the dividing or partitioning or points selection technique must be reliable that it should not affect the abstract of the data points.

2. PREVIOUS APPROACH
Many schemes have been proposed in past for predicting disease and parallelization of SVM some of the techniques that helps in development of our concepts in writing this paper are discussed here. For disease prediction Wei Yu*, Tiebin Liu, Rodolfo Valdez, Marta Gwinn, Muin J Khoury [11] used data from the 1999-2004 National Health and Nutrition Examination Survey (NHANES) to develop and validate SVM models for two classification schemes: Classification Scheme I (diagnosed or undiagnosed diabetes vs. pre-diabetes or no diabetes) and Classification Scheme II (undiagnosed diabetes or pre-diabetes vs. no diabetes). The SVM models were used to select sets of variables that would yield the best classification of individuals into these diabetes categories. Mohammed Khalilia, Sounak Chakraborty and Mihail Popescu [12] employed the National Inpatient Sample (NIS) data, which is publicly available through Healthcare Cost and Utilization Project (HCUP), to train random forest classifiers for disease prediction. Since the HCUP data is highly imbalanced, we employed an ensemble learning approach based on repeated random sub-sampling. This technique divides the training data into multiple sub-samples, while ensuring that each sub-sample is fully balanced. We compared

the performance of support vector machine (SVM), bagging, boosting and RF to predict the risk of eight chronic diseases. For parallel SVM the Yumao Lu and Vwani Roychowdhury [4] proposes A parallel support vector machine based on randomized sampling technique they modeled a new LP-type problem so that it works for general linear-nonseparable SVM training problems a unique priority based sampling mechanism is used so that we can prove an average convergence rate that is so far the fastest bounded convergence rate. Amit Maan et al.[5] introduce a distributed algorithm for solving large scale Support Vector Machines (SVM) problems. Their algorithm divides the training set into a number of processing nodes each running independently an SVM sub-problem associated with its subset of training data. The algorithm is a parallel (Jacobi) block-update scheme derived from the convex conjugate (Fenchel Duality) form of the original SVM problem. Each update step consists of a modified SVM solver running in parallel over the sub-problems followed by a simple global update. We derive bounds on the number of updates showing that the number of iterations (independent SVM applications on sub-problems) required to obtain a solution of accuracy ε is $O(\log(1/\varepsilon))$. The work proposed by Cheng-Tao Chu , Gary Bradski et el.[6] in their paper for a programming framework for processing with multicore processors in simple and unified way for machine learning to take advantage of the potential speed up. In paper, they develop a broadly applicable parallel programming method, one that is easily applied to many different learning algorithms. Our work is in distinct contrast to the tradition in machine learning of designing (often ingenious) ways to speed up a single algorithm at a time. Specifically, they show that algorithms that fit the Statistical Query model can be written in a certain "summation form," which allows them to be easily parallelized on multicore computers the proposed parallel speed up technique is tested on a variety of learning algorithms including locally weighted linear regression (LWLR), k-means, logistic regression (LR), naive Bayes (NB), SVM, ICA, PCA, Gaussian discriminant analysis (GDA), EM, and back propagation (NN) showing good results. To speed up the process of training SVM, another parallel methods have been proposed [7] by splitting the problem into smaller subsets and training a network to assign samples of different subsets. A parallel training algorithm on large-scale classification problems is proposed, in which multiple SVM classifiers are applied and may be trained in a distributed computer system. As an improvement algorithm of cascade SVM, the support vectors are obtained according to the data samples distance mean and the feedback is not the whole final output but alternating to avoid the problem that the learning results are subject to the distribution state of the data samples in different subsets. The experiment results on real-world text dataset show that this parallel SVM training algorithm is efficient and has more satisfying accuracy compared with standard cascade SVM algorithm in classification precision. The algorithm of Zanghirati and Zanni (2003) decomposes the SVM training problem into a sequence of smaller, though still dense, QP sub-problems. Zanghirati and Zanni implement the inner solver using a technique called variable projection method, which is able to work efficiently on relatively large dense inner problems, and is suitable for implementing in parallel. The performance of the inner QP solver was improved in Zanni et al. (2006). In the cascade algorithm introduced by Graf et al. (2005), the SVMs are layered. The support vectors given by the SVMs of one layer are combined to form the training sets of the next layer. The support vectors of the final layer are re-inserted into the training sets of the first layer at the next iteration, until the global KKT conditions are met. The authors show that this feedback loop corresponds to standard SVM training. The algorithm of Durdanovic et al. (2007), implemented in the Mild software, is a parallel implementation of the sequential minimal optimization.

3. SUPPORT VECTOR MACHINE

Support vector machine[14] is relatively new method of learning for two class classification problems the SVM maps the input vectors non linearly to a high dimensional feature space and build a linear decision boundary within this decision plane that will give the best generalization among all the hyper plane in the high dimensional feature space. The optimal hyper plane is defined as a linear decision function with the maximum distance between the vectors of two classes to build the optimal hyper plane only a small amount of training set examples need to be considered these examples are support vectors Support vector machine learning means to determine functions that can be used to classify data points the SVM learning method is based on so called reference data of given input output (training data). SVM is based on creating a hyper plane as the decision plane, which separates the positive (+1) and negative (-1) classes with the largest margin. An optimal hyper plane is the one with the maximum margin of separation between the two classes, where the margin is the sum of the distances from the hyper plane to the closest data points of each of the two classes. These closest data points are called Support Vectors (SVs). Given a set of training data D, a set of points of the type

$$D = \{(x_i, c_i) \mid x_i \in R^p, c_i \in \{-1, 1\}\} \qquad \text{(i)}$$

Where c_i is either 1 or -1 indicative of the class to which the point x_i belongs, the aim is to give a maximum margin hyperplane which divide points having $c_i = 1$ from those having $c_i = -1$. Any hyperplane can be constructed as a set of point x satisfying

$$w.x - b = 0. \qquad \text{(ii)}$$

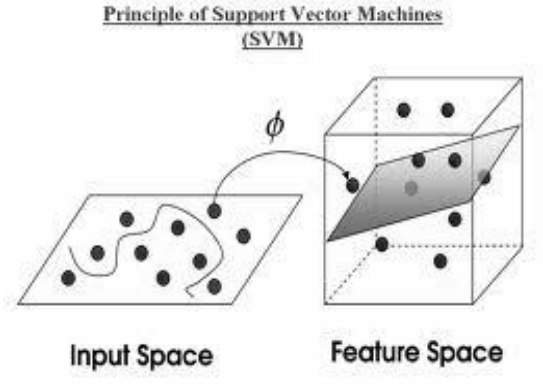

Fig1 : SVM Process Presentation

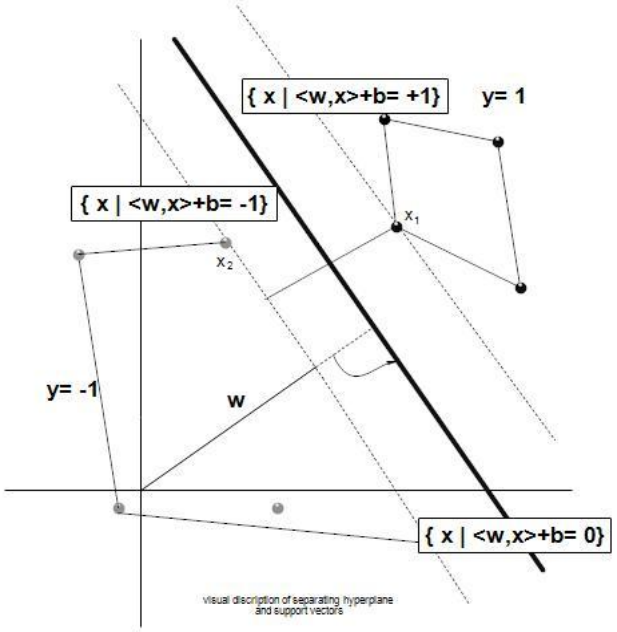

Fig2: Visual Description of Separating Hyperplane and Support Vectors

The vector **w** is a normal vector. We want to choose **w** and b to maximize the margin. These hyperplanes can be described by the following equations:

$$w.x - b = 1$$

$$w.x - b = -1$$

The margin is given by

$$m = 1/\|w\|^2$$

The dual of the SVM is shown to be the following optimization problem:

Maximize (in α_i)

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

Subject to

$$\alpha_i > 0 \text{ and } \sum_{i=1}^{n} \alpha_i y_i = 0$$

y_i indicates the class of an instance, there is a one-to-one association between each Lagrange multiplier α_i and each training example x_i. Once the Lagrange multipliers are determined, the normal vector **w** and the threshold b can be derived from the Lagranges multipliers as follow:

$$\vec{w} = \sum_{i=1}^{n} y_i a_i \vec{x_i}$$

$$b = \vec{w_k}.\vec{x_k} - y_k$$

for some $a_k > 0$. Not all data sets are linearly separable. There may be no hyperplane exist that separates the positive (+1) and negative (-1) classes. SVMs can be further generalized to non-linear classifiers. The output of a non-linear SVM is computed from the Lagrange multipliers as follow:

$$u = \sum_{i=1}^{n} y_i a_i K(X_i, X) + b$$

Where K is a kernel function that measures the similarity or distance between the input vector X_i and the stored training vector X.

4. PARALLEL SVM

Penalization of any algorithm is a concept to arrange or partition the process of an algorithm such that it can be parallel processed on cluster of computers. In context of this particular paper we denoting Parallel SVM as the concept of partitioning a large training dataset into small data chunks and process each chunk in parallel utilizing the resources of a cluster of computers.It's already clear from previous sections that training SVMs is computationally intensive and increases dramatically as the size of a training dataset increases. A SVM kernel usually involves an algorithmic complexity of $O(m^2n)$, where n is the dimension of the input and m represents the training instances [3]. The computation time in SVM training is quadratic in terms of the number of training instances. Hence parallel approximate implementation to speed up SVM training on today's distributed computing infrastructures has proposed although the Parallel SVM is the sole solution to speed up SVMs. Algorithmic approaches such as (Lee & Mangasarian, 2001 [8]; Tsang et al., 2005; Joachims, 2006; Chu et al.,2006) [9], can be more effective when memory is not a constraint or kernels are not used.

5. K MEANS CLUSTERING

In statistics and data mining, ***k*-means clustering** is a method of cluster analysis which aims to partition *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean. This results into a partitioning of the data space into Voronoi cells. The problem is computationally difficult (NP-hard), however there are efficient heuristic algorithms that are commonly employed that converge fast to a local optimum. These are usually similar to the expectation-maximization algorithm for mixtures of Gaussian distributions via an iterative refinement approach employed by both algorithms. Additionally, they both use cluster centers to model the data, however *k*-means clustering tends to find clusters of comparable spatial extend, while the expectation maximization mechanism allows clusters to have different shapes [10].

Given a set of observations ($\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$), where each observation is a *d*-dimensional real vector, *k*-means clustering aims to partition the *n* observations into *k* sets ($k \le n$) $\mathbf{S} = \{S_1, S_2, \ldots, S_k\}$ so as to minimize the within-cluster sum of squares (WCSS):

$$\arg\min s \sum_{i=1}^{k} \sum_{x_j \in s_i} \left\| x_j - \mu_i \right\|^2$$

where $\boldsymbol{\mu}_i$ is the mean of points in S_i

6. PROPOSED ALGORITHM

Penalization of SVM has been already discussed in section 4. The proposed algorithm also follows the concept developed in that section. The proposed algorithm finds the minimum numbers of data points which represents the abstracts of large dataset this is performed by firstly partitioning the data points into n numbers of clusters the n depends upon the size of dataset, number of available processors, computational power of processors and available memory. The first level partitioning is performed by K means clustering, but before using clustering the data points which does not participate or influence the partitioning planes are removed. This process also helps in balancing the both class data. The complete step by step description of algorithm is given below

Algorithm in steps:

1. Let the positive and negative class datasets be D_P and D_N
2. Choose the set with minimum size let it is D_P
3. Calculate its centre point C_P
4. Calculate the distance of all vectors of (D_N) from C_P
5. Choose the n minimum distance vectors form D_N dataset, where n is the size of D_P
6. Name it D_{N1} now make a new data set D_{new} = (D_P + D_N)
7. Divide the D_{new} in N sections by K mans clustering
8. Calculate the Support Vectors of each sections S_{ij} with their Class L_{ij} , where i = {1,2,3…..N} and j = {P, N}
9. Train the final SVM using S_{ij} and L_{ij}.

Explanation of the algorithm

1. The data set for training having two classes only and having unequal sizes.
2. The steps 2, 3, 4 and 5 are used to eliminate the non useful vectors from dataset and also balance the classification dataset.
3. The purpose of k means is to divide data for parallel processing it divides the dataset in most similar N sets which are most difficult to classify when grouped together.
4. The calculation of support vectors from each section provides the abstracted information of all vector of that section with only a fewer vectors which reduces the load for final classifier. Creation of final classifier for future classification

7. FLOW CHART

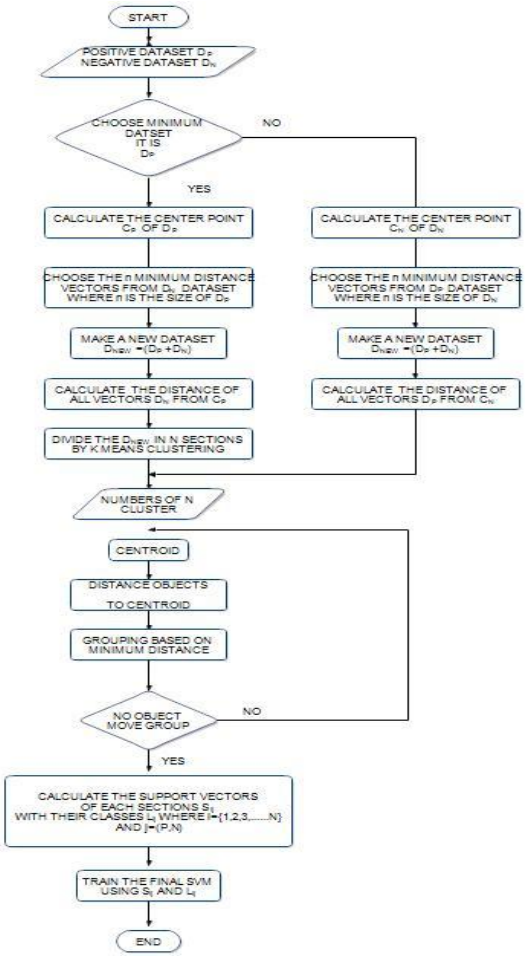

Fig3. Flow Chart of Parallel SVM with K Means Clustering

8. SIMULATION RESULTS

The proposed algorithm is developed in MATLAB 7.12.7, R2011B and the simulation results are obtained by running it on intel celeron with 3 GB of RAM. The dataset is taken from " S. S. Medical College, Rewa, Master Chart (Study Group)" and following parameters are taken for formation of vectors

1. Age
2. Sex
3. Family history of diabetes
4. BMI (body mass index)
5. SBP (Systolic blood pressure)
6. DBP (Diastolic blood pressure)
7. FSUG (Fasting blood sugar)

For number of cluster variations

Table 1 Simple SVM

Data Size	Cluster	TPR	TNR	FPR	FNR	Accuracy	Time (sec.)
50	-	0.59	0.92	0.07	0.40	0.75	1.16

Table 2. Parallel SVM

Data Size	Cluster	TPR	TNR	FPR	FNR	Accuracy	Time (sec.)
50	2	0.55	0.92	0.07	0.44	0.74	0.28
50	3	0.48	0.92	0.07	0.51	0.70	0.37
50	4	0.48	0.92	0.07	0.51	0.70	0.40
50	5	0.25	0.92	0.07	0.74	0.59	0.42
50	6	0.25	0.92	0.07	0.74	0.59	0.44

Table 3. Simple SVM

Data Size	Cluster	TPR	TNR	FPR	FNR	Accuracy	Time (sec.)
100	-	0.59	0.88	0.11	0.40	0.74	1.32

Table 4.Parallel SVM

Data Size	Cluster	TPR	TNR	FPR	FNR	Accuracy	Time (sec.)
100	2	0.59	0.92	0.07	0.40	0.75	0.31
100	3	0.59	0.92	0.07	0.40	0.75	0.27
100	4	0.59	0.92	0.07	0.40	0.75	0.31
100	5	0.37	0.92	0.07	0.62	0.64	0.32
100	6	0.37	0.92	0.07	0.62	0.64	0.36

Table 5. Simple SVM

Data Size	Cluster	TPR	TNR	FPR	FNR	Accuracy	Time (sec.)
200	-	0.62	0.92	0.07	0.37	0.77	2.10

Table 6.Parallel SVM

Data Size	Cluster	TPR	TNR	FPR	FNR	Accuracy	Time (sec.)
200	2	0.62	0.92	0.07	0.37	0.77	0.30
200	3	0.62	0.92	0.07	0.37	0.77	0.31
200	4	0.62	0.92	0.07	0.37	0.77	0.33
200	5	0.62	0.92	0.07	0.37	0.77	0.36
200	6	0.62	0.92	0.07	0.37	0.77	0.39

9. ANALYSIS

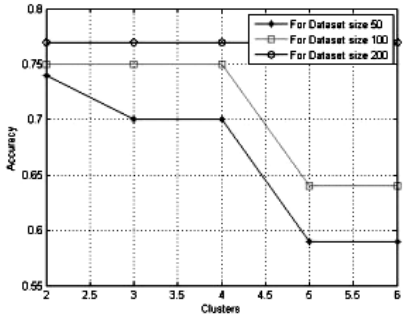

Fig4: Clusters vs Accuracy

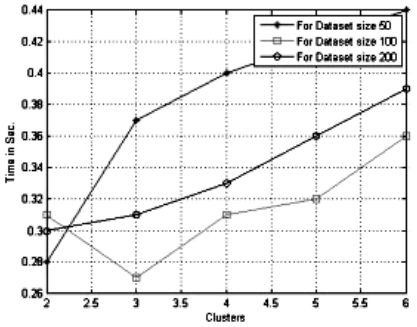

Fig5: Clusters vs Time in sec

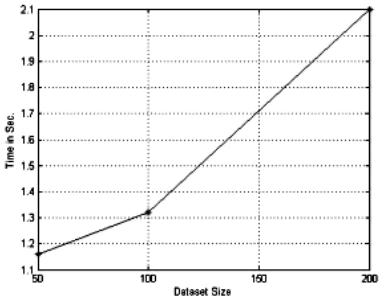

Fig3: Datset size vs Time in sec

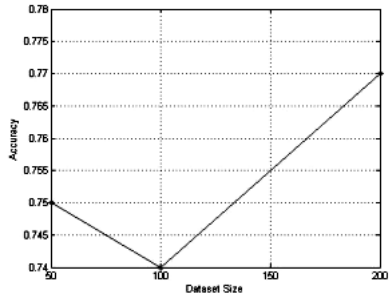

Fig4: Dataset size vs Accuracy

10. CONCLUSION

The simulation results shows that proposed algorithm takes only 1/3 of time taken by normal SVM for training and this result is for single machine so expected results for multi machine case should be dropped by n times where n is number of machines. The proposed method also maintained the approximately same accuracy when compared with normal SVM, although it shows that dividing data into larger number of clusters decreases the accuracy but it could be controlled by selecting proper starting point for K means clustering we leaved this work for future.

11. AKNOWLEDGEMENT

Our sincere thanks to the those who contribute there valuable guidance for our work

12. REFERENCES

[1] Horváth (2003) in Suykens et al. p 392

[2] Kristian Woodsend and Jacek Gondzio"*Hybrid MPI/OpenMP Parallel Linear Support Vector Machine Training"* Journal of Machine Learning Research 10 (2009) 1937-1953.

[3] Nasullah Khalid Alham brunel university PHD thesis on *"Parallelizing Support Vector Machines for Scalable Image Annotation"*

[4] Yumao Lu and Vwani Roychowdhury *"Parallel Randomized Support Vector Machine"* PAKDD 2006, LNAI 3918, pp. 205–214, 2006.

[5] Tamir Hazan Amit Man Amnon Shashua "*A Parallel Decomposition Solver for SVM: Distributed Dual Ascend using Fenchel Duality"*

[6] Cheng-Tao Chu, Sang Kyun Kim, Yi-An Lin, YuanYuan Yu, Gary Bradski, Andrew Y. Ng, Kunle Olukotun "*Map-Reduce for Machine Learning on Multicore"* CS. Department, Stanford University 353 Serra Mall, Stanford University, Stanford CA 94305-9025 Rexee Inc.

[7] "Jian-Pei Zhang; Zhong-Wei Li; Jing Yang; "*A parallel SVM training algorithm on large-scale classification problems"* Coll. of Comput. Sci. & Technol., Harbin Eng. Univ., China IEEE Machine Learning and Cybernetics, 2005. Proceedings of 2005.

[8] Lee, Y.-J., & Mangasarian, O. L. (2001). *"Rsvm: Reduced support vector machines"*. First SIAM International Conference on Data Mining. Chicago.

[9] Joachims, T. (1998*). "Making large-scale svm learning practical. Advances in Kernel Methods Support Vector Learning."*

[10] K-means clustering available at: http://en.wikipedia.org/wiki/Kmeans_clustring

[11] Wei Yu, Tiebin Liu, Rodolfo Valdez, Marta Gwinn, Muin J Khoury *"Application of support vector machine modeling for prediction of common diseases: the case of diabetes and prediabetes".* BMC Medical Informatics and Decision Making 2010, 10:16

[12] Mohammed Khalilia, Sounak Chakrabortyand Mihail Popescu *"Predicting disease risks from highly imbalanced data using random forest"* BMC Medical Informatics and Decision Making 2011,11:51