Abstract—Software Effort Estimation is important for both the software development organization as well as the organization for which it is made. SVM and kNN are computer-based techniques to estimate effort. kNN and SVM based approach could serve as an economical, automatic tool to generate ranking of software by formulating the relationship based on its training. When there are some data points that belong to one of two classes then the goal is to decide about the class of new data point.

Keywords— Software Effort Estimation, Support Vector machines (SVM), Nearest Neighbor Approach (kNN).

I. INTRODUCTION

SOFTWARE Effort Estimation is important for both the software development organization as well as the organization for which it is made. If estimates are accurate only then good deal can be made otherwise software developers can commit wrong promises. Accurate cost estimation helps to classify and prioritize, development projects with respect to an overall business plan. It can be used to determine what resources to commit to the project and how well these resources will be used. It can be used to assess the impact of changes and support re-planning. Projects can be easier to manage and control, when resources are better matched to real needs. Customers expect actual development costs to be in line with estimated costs.

Software cost estimation involves the determination of effort (in person-months), project duration (in calendar time) and cost (in dollars). A good software cost estimate in literature [1]:

- It is conceived and supported by the project manager and the development team.
- It is accepted by all stakeholders as realizable.
- It is based on a well-defined software cost model with a credible basis.
- It is based on a database of relevant project experience (similar processes, similar technologies, similar environments, similar people and similar requirements).
- It is defined in enough detail so that its key risk areas are understood and the probability of success is objectively assessed.

Software estimation is difficult because it is highly complex process, premature estimate requests and need to be calibrate to local environments and dynamic evolution environment, [7].

II. SOFTWARE EFFORT ESTIMATION

According to K.Smith, et.al. [2] there are four task assignment factors, team size, concurrency, intensity, and fragmentation that influence the software effort. According to Boehm, there are three main methodology categories of software cost estimation [5]:

a) Estimation by consulting experts.

b) Estimation by analogy based on estimation by using past projects sharing similar characteristics.

c) Estimation by Algorithmic modeling.

Normally, estimation is performed using only human expertise [3], [4], but recently attention has turned to a variety of computer-based learning techniques. SVM and kNN based approach could serve as an economical, automatic tool to generate ranking of software by formulating the relationship based on its training. Suppose some given data points each belong to one of two classes, and the goal is to decide about the class of new data point.

III. SUPPORT VECTOR MACHINE (SVM)

SVM algorithm was introduced in COLT-92 by Boser, Guyon & Vapnik. SVM or a support vector machine constructs a hyperplane or set of hyperplanes in a high or infinite dimensional space, which can be used for classification, regression, or other tasks. The main purpose of SVM is to differentiate two different types of samples, meanwhile insuring classification of the largest interval and the smallest error rate, [6] . SVM has a strong advantage in assessment, through selecting a appropriate parameter C , ε , constructing a greatest and optimal hyper-plane to contains as much support vector as possible, thus avoiding the local optimal solution problem of neural network and ensuring the generalization ability of SVM, [8]. A step in SVM classification involves identification as which are intimately connected to the known classes. This is called feature selection or feature extraction. Feature selection and SVM classification together have a use even when prediction of unknown samples is not necessary. They can be used to identify key sets which are involved in whatever processes distinguish the classes. The major strengths of SVM are the...
training is relatively easy. No local optimal, unlike in neural networks. It scales relatively well to high dimensional data and the trade-off between classifier complexity and error can be controlled explicitly. In the case of support vector machines, a data point is viewed as a p-dimensional vector (a list of p numbers), and we want to know whether we can separate such points with a (p – 1)-dimensional hyperplane. This is called a linear classifier. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training data points of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier. SVM can estimate the functional dependence of the variable y on a set of independent variables x. It presumes, that the relationship between the independent and dependent variables is given by a deterministic function f plus the addition of some additive noise: y = f(x) + noise.

IV. K-NEAREST NEIGHBORS’ (KNN)

The k-nearest neighbors’ algorithm (k-NN) is a method for classifying objects based on closest training examples in the feature space. k-NN is a type of instance-based learning, or lazy learning where the function is only approximated locally and all computation is deferred until classification. The k-nearest neighbor algorithm is amongst the simplest of all machine learning algorithms: an object is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of its nearest neighbor. The same method can be used for regression, by simply assigning the property value for the object to be the average of the values of its k nearest neighbors. It can be useful to weight the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. A common weighting scheme is to give each neighbor a weight of 1/d, where d is the distance to the neighbor. This scheme is a generalization of linear interpolation.). The neighbors are taken from a set of objects for which the correct classification (or, in the case of regression, the value of the property) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required. The k-nearest neighbor algorithm is sensitive to the local structure of the data. Nearest neighbor rules in effect compute the decision boundary in an implicit manner. It is also possible to compute the decision boundary itself explicitly, and to do so in an efficient manner so that the computational complexity is a function of the boundary complexity. The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples. In the classification phase, k is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the k training samples nearest to that query point.

Usually Euclidean distance is used as the distance metric; however this is only applicable to continuous variables. In cases such as text classification, another metric such as the overlap metric (or Hamming distance) can be used. Predictions are based on averaging the outcomes from the k-nearest neighbours. Value of k is significant factor, and a smoothing parameter that can manoeuvre the predictions. The value of k should be precise so as to achieve a balance between bias and variance of the model.

V. CONCLUSION

We have gone through the survey of the SVM and kNN models for various applications and we conclude that most of the software quality evaluation problems the performance of SVM model is better than the kNN approach. Hence, we will also try Software Effort Estimation using SVM.

REFERENCES