A Project Report

On

SOLVING AI PROBLEMS USING MACHINE LEARNING

Submitted in Partial Fulfillment of the Requirements for the award of the degree

Of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

BY

MOHAMED KHALID 160313737019

MOHD JAWAD 160313737038

MOHAMMED OMAR SHAKEEL 160313737045

Under the guidance of

**FEROZ AMER**

Associate Professor

Department of I.T

DCET, Hyderabad.

**Department of Information Technology**

**Deccan College of Engineering and Technology**

**(Affiliated to Osmania University)**

**Hyderabad**

**DECLARATION**

We hereby declare that the project work entitled “Solving AI Problems Using Machine Learning” submitted to the OSMANIA UNIVERSITY, is a record of original work done by us under the guidance of “Mr. Feroz Amer, Associate Professor, Dept. of Information Technology, DECCAN COLLEGE OF ENGINEERING AND TECHNOLOGY”. The project reports are based on the project work done entirely by us and not copied from any other source.

MOHAMED KHALID 160313737019

MOHD JAWAD 160313737038

MOHAMMED OMAR SHAKEEL 160313737045

**ACKNOWLEDGEMENT**

We express our deep gratitude to our project guide MR. FEROZ AMER, for his extreme support and guidance in making out of this project successfully.

It is our pleasure to express our whole hearted thanks to DR. SYED RAZIUDDIN, H.O.D, IT Department, Deccan College of Engineering & Technology, for providing necessary facilities to undergo this project.

We are also thankful to all the staff members of IT Department and also our friends who helped us in this project.

With pleasure we would like to take this opportunity to thank our families for their encouragement and support to complete this project work.

MOHAMED KHALID MOHD JAWAD MOHAMMED OMAR SHAKEEL

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Figure Name** | **Page No.** |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 9 |  |  |
| 10 |  |  |
| 11 |  |  |
| 12 |  |  |
| 13 |  |  |
| 14 |  |  |
| 15 |  |  |
| 16 |  |  |
| 17 |  |  |
| 18 |  |  |
| 19 |  |  |
| 20 |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Table Name** | **Page No.** |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 9 |  |  |
| 10 |  |  |
| 11 |  |  |
| 12 |  |  |
| 13 |  |  |
| 14 |  |  |
| 15 |  |  |
| 16 |  |  |
| 17 |  |  |
| 18 |  |  |
| 19 |  |  |
| 20 |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**ABSTRACT**

Machine learning is a tool for data analysis.  It Uses algorithms that iteratively learn from data, machine learning allows computers to find hidden patterns without being explicitly programmed where to look.

**Customer Segmentation**:

Given the pattern of behavior by a user during a trial period and the past behaviors of all users, identify those users that will convert to the paid version of the product and those that will not. A model of this decision problem would allow a program to trigger customer interventions to persuade the customer to convert early.

**Digit Recognition**:

Given a zip codes hand written on envelops, identify the digit for each hand written character. A model of this problem would allow a computer program to read and understand handwritten zip codes and sort envelops by geographic region.

**Shape Detection**:

Given a user hand drawing a shape on a touch screen and a database of known shapes, determine which shape the user was trying to draw. A model of this decision would allow a program to show the platonic version of that shape the user drew to make crisp diagrams.

.

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Title** | **Page No.** |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 9 |  |  |
| 10 |  |  |
| 11 |  |  |
| 12 |  |  |
| 13 |  |  |
| 14 |  |  |
| 15 |  |  |
| 16 |  |  |
| 17 |  |  |
| 18 |  |  |
| 19 |  |  |
| 20 |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**1. Introduction**

**1.1 What is machine learning?**

Machine learning is a tool for data analysis.  It Uses algorithms that iteratively learn from data, machine learning allows computers to find hidden patterns without being explicitly programmed where to look.

**1.2 Why do we use machine Learning?**

All of the things like growing volumes and varieties of available data , computational processing that is more cheaper and powerful and affordable data storage  mean it's possible to quickly and automatically produce models that can analyze bigger, more complex data and deliver faster, more accurate results – even on a very large scale. And by building precise models, an organization has a better chance of identifying profitable opportunities – or avoiding unknown risks.

**1.3 Why is it more reliable now than ever before?**

Because of new computing technologies, machine learning today is not like machine learning of the past. It was born from pattern recognition and the theory that computers can learn without being programmed to perform specific tasks; researchers interested in artificial intelligence wanted to see if computers could learn from data. The iterative aspect of machine learning is important because as models are exposed to new data, they are able to independently adapt. They learn from previous computations to produce reliable, decisions and results.

There were many machine learning algorithms around for while. Applying these algorithms iteratively on data and coming to conclusions with very high confidence is a new improvement

Here are a few widely publicized examples of machine learning applications you may be familiar with:

* The heavily hyped, self-driving Google car? The essence of machine learning.
* Online recommendation offers such as those from Amazon and Netflix? Machine learning applications for everyday life.
* Fraud detection? One of the more obvious, important uses in our world today.

**1.4 Different types of machine learning methods:**

There are several machine learning methods being used, but two of them are widely adapted (supervised and unsupervised Learning).

Some of the machines learning methods are:

1. Supervised Learning
2. Unsupervised Learning
3. Semi supervised Learning
4. Reinforcement Learning

**1.4.1 Supervised Learning:**

This type of learning mainly focuses on learning some large amount of training data and coming up with the function f(x) = y, where x is the input vector and y is the label or class to classify the data points. For example data points can be classified as either "Pass" or "Fail".

Through methods like classification, regression, prediction , supervised learning finds patterns to label  unlabeled data points. For example it can anticipate when a credit card transaction is fraudulent.

**1.4.2 Unsupervised Learning:**

In this type of learning there is not always a "right answer", in fact there may be many right answers. In this type of learning the algorithm must figure out some patterns in the data set,

The goal is to find some structure in the data. For example  Clustering is a type of unsupervised learning since there may be more than one right answer

**1.5 Types of Machine Learning Problems**:

There are common classes of problems in Machine Learning. When one finds a problem to be a machine learning problem, the first step is to decide on what type of machine learning problem they are dealing with. The problem classes below are archetypes for most of the problems we refer to in Machine Learning.

**1.5.1 Classification**:

|  |  |
| --- | --- |
| Data is labeled meaning it is assigned a class, for example spam/non-spam or fraud/non-fraud. The decision being modelled is to assign labels to new unlabeled pieces of data. This can be thought of as a discrimination problem, modelling the differences or similarities between groups. | 1.PNG |

**1.5.2 Regression:**

|  |  |
| --- | --- |
| Data is labeled with a real value rather than a label. Examples that are easy to understand are time series data like the price of a stock over time, The decision being modeled is what value to predict for new unpredicted data. | **C:\Users\Farooqi\Desktop\2.PNG** |

**1.5.3 Clustering:**

|  |  |
| --- | --- |
| Data is not labeled or classified, but can be divided into groups based on similarity and other measures of natural structure in the data. Data points that belong to the same cluster are similar to each other and data points from different cluster differ. An example is shown in the below diagram where a data set is divided into five clusters based on distance using k-means clustering algorithm. | C:\Users\Farooqi\Desktop\3.PNG |

**1.5.4 Rule Extraction**:

Data is used as the basis for the extraction of propositional rules. These rules are typically not directed, meaning that the methods discover statistically supportable relationships between attributes in the data, not necessarily involving something that is being predicted. An example is the discovery of the relationship between the purchase of pencils and erasers.

**2. LITERATURE SURVEY**

Machine learning is a type of artificial intelligence that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of computer programs that can change when exposed to new data. The process of machine learning is similar to that of data mining. It is useful to tour the main algorithms in the field to get a feeling of what methods are available.

There are so many algorithms available that it can feel overwhelming when algorithm names are thrown around and you are expected to just know what they are and where they fit. There are two ways to think about and categorize the algorithms you may come across in the field.

* The first is a grouping of algorithms by the learning style.
* The second is a grouping of algorithms by similarity in form or function (like grouping similar animals together).

**2.1 Algorithms Grouped by Learning Style:**

There are different ways an algorithm can model a problem based on its interaction with the experience or environment or whatever we want to call the input data. It is popular in machine learning and artificial intelligence textbooks to first consider the learning styles that an algorithm can adopt.

There are only a few main learning styles or learning models that an algorithm can have and we’ll go through them here with a few examples of algorithms and problem types that they suit. This taxonomy or way of organizing machine learning algorithms is useful because it forces you to think about the roles of the input data and the model preparation process and select one that is the most appropriate for your problem in order to get the best result.

The three different learning styles in machine learning algorithms are:

**2.1.1 Supervised Learning**

|  |  |
| --- | --- |
| Input data is called training data and has a known label or result such as spam/not-spam or a stock price at a time. A model is prepared through a training process in which it is required to make predictions and is corrected when those predictions are wrong. The training process continues until the model achieves a desired level of accuracy on the training data. Example problems are classification and regression. Example algorithms include Logistic Regression and the Back Propagation Neural Network. | Supervised Learning Algorithms |

**2.1.2 Unsupervised Learning**

|  |  |
| --- | --- |
| Input data is not labeled and does not have a known result. A model is prepared by deducing structures present in the input data. This may be to extract general rules. It may be through a mathematical process to systematically reduce redundancy, or it may be to organize data by similarity. Example problems are clustering, dimensionality reduction and association rule learning. Example algorithms include: the Apriori algorithm and k-Means. | Unsupervised Learning Algorithms |

**2.1.3 Semi-Supervised Learning**

|  |  |
| --- | --- |
| Input data is a mixture of labeled and unlabelled examples. There is a desired prediction problem but the model must learn the structures to organize the data as well as make predictions. Example problems are classification and regression. Example algorithms are extensions to other flexible methods that make assumptions about how to model the unlabeled data. | [Semi-supervised Learning Algorithms](http://3qeqpr26caki16dnhd19sv6by6v.wpengine.netdna-cdn.com/wp-content/uploads/2013/11/Semi-supervised-Learning-Algorithms.png) |

**2.1.4 Overview**

When crunching data to model business decisions, you are most typically using supervised and unsupervised learning methods. A hot topic at the moment is semi-supervised learning methods in areas such as image classification where there are large datasets with very few labeled examples.

**2.2 Algorithms Grouped By Similarity**

Algorithms are often grouped by similarity in terms of their function (how they work). For example, tree-based methods, and neural network inspired methods. This is a useful grouping method, but it is not perfect. There are still algorithms that could just as easily fit into multiple categories like Learning Vector Quantization that is both a neural network inspired method and an instance-based method. There are also categories that have the same name that describe the problem and the class of algorithm such as Regression and Clustering. We could handle these cases by listing algorithms twice or by selecting the group that subjectively is the “best” fit. I like this latter approach of not duplicating algorithms to keep things simple.

**2.2.1 Regression Algorithms**

|  |  |
| --- | --- |
| Regression is concerned with modeling the relationship between variables that is iteratively refined using a measure of error in the predictions made by the model. Regression methods are a workhorse of statistics and have been co-opted into statistical machine learning. This may be confusing because we can use regression to refer to the class of problem and the class of algorithm. Really, regression is a process. | Regression Algorithms |

The most popular regression algorithms are:

* Ordinary Least Squares Regression (OLSR)
* Linear Regression
* Logistic Regression
* Stepwise Regression
* Multivariate Adaptive Regression Splines (MARS)
* Locally Estimated Scatter plot Smoothing (LOESS)

**2.2.2 Instance-based Algorithms**

|  |  |
| --- | --- |
| Instance-based learning model is a decision problem with instances or examples of training data that are deemed important or required to the model. Such methods typically build up a database of example data and compare new data to the database using a similarity measure in order to find the best match and make a prediction. For this reason, instance-based methods are also called winner-take-all methods and memory-based learning. Focus is put on the representation of the stored instances and similarity measures used between instances. | Instance-based Algorithms |

The most popular instance-based algorithms are:

* k-Nearest Neighbor (KNN)
* Learning Vector Quantization (LVQ)
* Self-Organizing Map (SOM)
* Locally Weighted Learning (LWL)

**2.2.3 Regularization Algorithms**

|  |  |
| --- | --- |
| An extension made to regression methods, that penalizes models based on their complexity, favoring simpler models that are also better at generalizing. These algorithms are listed here because they are popular, powerful and generally simple modifications made to other methods. | Regularization Algorithms |

The most popular regularization algorithms are:

* Ridge Regression
* Least Absolute Shrinkage and Selection Operator (LASSO)
* Elastic Net
* Least-Angle Regression (LARS)

**2.2.4 Decision Tree Algorithms:**

|  |  |
| --- | --- |
| Decision tree methods construct a model of decisions made based on actual values of attributes in the data. Decisions fork in tree structures until a prediction decision is made for a given record. Decision trees are trained on data for classification and regression problems. Decision trees are often fast and accurate and a big favorite in machine learning. | Decision Tree Algorithms |

The most popular decision tree algorithms are:

* Classification and Regression Tree (CART)
* Iterative Dichotomiser 3 (ID3)
* C4.5 and C5.0 (different versions of a powerful approach)
* Chi-squared Automatic Interaction Detection (CHAID)
* Decision Stump
* M5
* Conditional Decision Trees

**2.2.5 Bayesian Algorithms:**

|  |  |
| --- | --- |
| Bayesian methods are those that explicitly apply Bayes’ Theorem for problems such as classification and regression. | Bayesian Algorithms |

The most popular Bayesian algorithms are:

* Naive Bayes
* Gaussian Naive Bayes
* Multinomial Naive Bayes
* Averaged One-Dependence Estimators (AODE)
* Bayesian Belief Network (BBN)
* Bayesian Network (BN)

**2.2.6 Clustering Algorithms:**

|  |  |
| --- | --- |
| Clustering, like regression, describes the class of problem and the class of methods. Clustering methods are typically organized by the modeling approaches such as centroid-based and hierarchal. All methods are concerned with using the inherent structures in the data to best organize the data into groups of maximum commonality. | Clustering Algorithms |

The most popular clustering algorithms are:

* k-Means
* k-Medians
* Expectation Maximisation (EM)
* Hierarchical Clustering

**2.2.7 Association Rule Learning Algorithms:**

|  |  |
| --- | --- |
| Association rule learning methods extract rules that best explain observed relationships between variables in data. These rules can discover important and commercially useful associations in large multidimensional datasets that can be exploited by an organization. | Assoication Rule Learning Algorithms |

The most popular association rule learning algorithms are:

* Apriori algorithm
* Eclat algorithm

**2.2.8 Artificial Neural Network Algorithms:**

|  |  |
| --- | --- |
| Artificial Neural Networks are models that are inspired by the structure and/or function of biological neural networks. They are a class of pattern matching that are commonly used for regression and classification problems but are really an enormous subfield comprised of hundreds of algorithms and variations for all manner of problem types. The most popular artificial neural network algorithms are: | Artificial Neural Network Algorithms |

The most popular association rule learning algorithms are:

* Perceptron
* Back-Propagation
* Hopfield Network
* Radial Basis Function Network (RBFN)

**2.2.9 Deep Learning Algorithms:**

|  |  |
| --- | --- |
| Deep Learning methods are a modern update to Artificial Neural Networks that exploit abundant cheap computation. They are concerned with building much larger and more complex neural networks and, as commented on above, many methods are concerned with semi-supervised learning problems where large datasets contain very little labeled data. | Deep Learning Algorithms |

The most popular deep learning algorithms are:

* Deep Boltzmann Machine (DBM)
* Deep Belief Networks (DBN)
* Convolutional Neural Network (CNN)
* Stacked Auto-Encoders

**2.2.10 Dimensionality Reduction Algorithms:**

|  |  |
| --- | --- |
| Like clustering methods, dimensionality reduction seek and exploit the inherent structure in the data, but in this case in an unsupervised manner or order to summarize or describe data using less information. This can be useful to visualize dimensional data or to simplify data which can then be used in a supervised learning method. | Dimensional Reduction Algorithms |

Many of these methods can be adapted for use in classification and regression.

* Principal Component Analysis (PCA)
* Principal Component Regression (PCR)
* Partial Least Squares Regression (PLSR)
* Sammon Mapping
* Multidimensional Scaling (MDS)
* Projection Pursuit
* Linear Discriminant Analysis (LDA)
* Mixture Discriminant Analysis (MDA)
* Quadratic Discriminant Analysis (QDA)
* Flexible Discriminant Analysis (FDA)

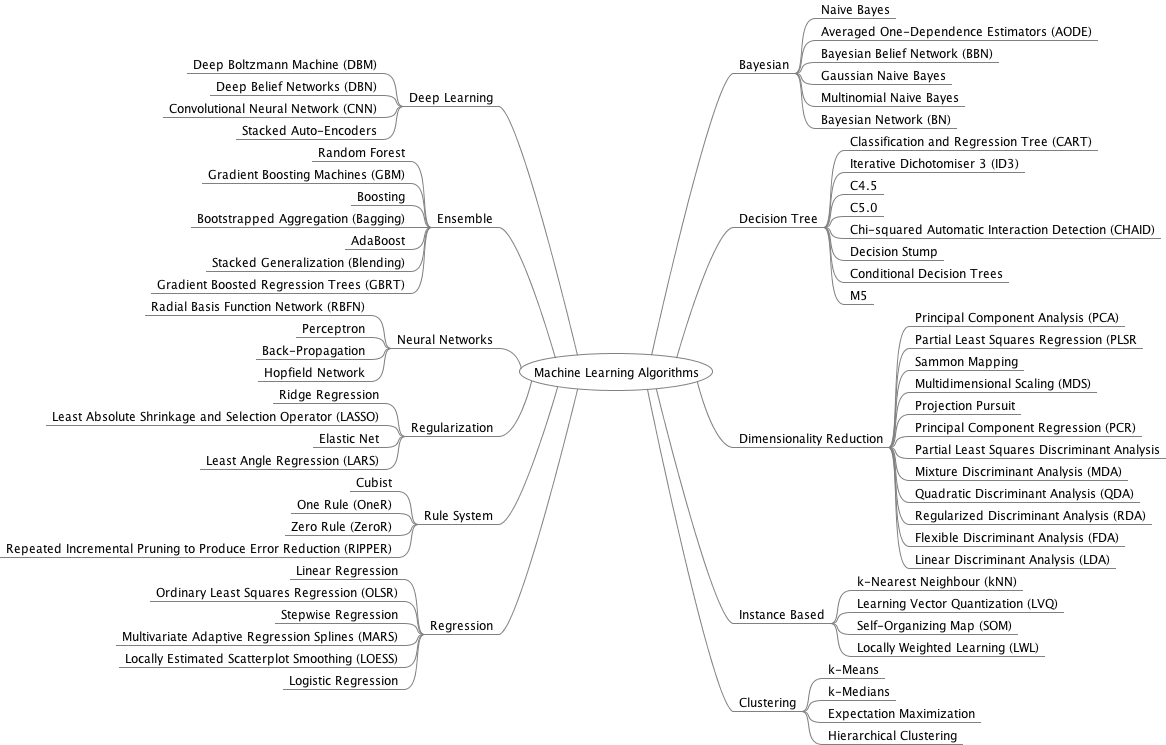
**2.2.11 Ensemble Algorithms:**

|  |  |
| --- | --- |
| Ensemble methods are models composed of multiple weaker models that are independently trained and whose predictions are combined in some way to make the overall prediction. Much effort is put into what types of weak learners to combine and the ways in which to combine them. | Ensemble Algorithms |

This is a very powerful class of techniques and as such is very popular.

* Boosting
* Bootstrapped Aggregation (Bagging)
* AdaBoost
* Stacked Generalization (blending)
* Gradient Boosting Machines (GBM)
* Gradient Boosted Regression Trees (GBRT)
* Random Forest

Here is a mind map of most of the important Machine learning algorithms



**2.3 Customer Segmentation**:

Given the pattern of behavior by a user during a trial period and the past behaviors of all users, identify those users that will convert to the paid version of the product and those that will not. A model of this decision problem would allow a program to trigger customer interventions to persuade the customer to convert early. Customer segmentation is the practice of dividing a customer base into groups of individuals that are similar in specific ways relevant to marketing, such as age, gender, interests and spending habits. Customer segmentation, also called consumer segmentation or client segmentation, procedures include:

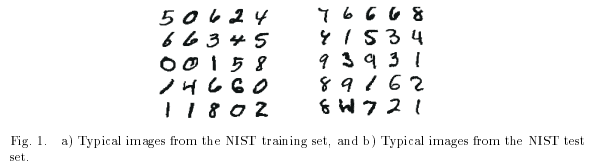
* Deciding what data will be collected and how it will be gathered
* Collecting data and integrating data from various sources
* Developing methods of [data analysis](http://searchsqlserver.techtarget.com/definition/data-mining) for segmentation
* Establishing effective communication among relevant business units (such as marketing and customer service) about the segmentation
* Implementing applications to effectively deal with the data and respond to the information it provides

Companies employing customer segmentation operate under the fact that every customer is different and that their marketing efforts would be better served if they target specific, smaller groups with messages that those consumers would find relevant and lead them to buy something. Companies also hope to gain a deeper understanding of their customers' preferences and needs with the idea of discovering what each segment finds most valuable to more accurately tailor marketing materials toward that segment. Customer segmentation relies on identifying key differentiators that divide customers into groups that can be targeted. Information such as a customers' [demographics](http://whatis.techtarget.com/definition/demographic) (age, race, religion, gender, family size, ethnicity, income, education level), geography (where they live and work), psychographic (social class, lifestyle and personality characteristics) and behavioral (spending, consumption, usage and desired benefits) tendencies are taken into account when determining customer segmentation practices.

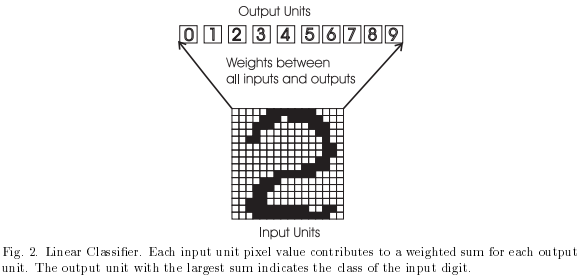
By enabling companies to target specific groups of customers, a customer segmentation model allows for the effective allocation of marketing resources and the maximization of [cross](http://searchcrm.techtarget.com/definition/cross-sell)- and [up-selling](http://searchcrm.techtarget.com/definition/up-sell) opportunities. When a group of customers is sent personalized messages as part of a marketing mix that is designed around their needs, it's easier for companies to send those customers special offers meant to encourage them to buy more products. Customer segmentation can also improve customer service and assist in customer loyalty and retention. As a by-product of its personalized nature, marketing materials sent out using customer segmentation tend to be more valued and appreciated by the customer who receives them as opposed to impersonal brand messaging that doesn't acknowledge purchase history or any kind of customer relationship. Other benefits of customer segmentation include staying a step ahead of competitors in specific sections of the market and identifying new products that existing or potential customers could be interested in or improving products to meet customer expectations. Not only do companies strive to divide their customers into measurable segments according to their needs, behaviors or demographics but they also aim to determine the profit potential of each segment by analyzing its revenue and cost impacts. Value-based segmentation evaluates groups of customers in terms of the revenue they generate and the costs of establishing and maintaining relationships with them. It also helps companies determine which segments are the most and least profitable so that they can adjust their marketing budgets accordingly. Customer segmentation can have a great effect on customer management in that, by dividing customers into different groups that share similar needs, the company can market to each group differently and focus on what each kind of customer needs at any given moment. Large or small, niche customer segments can be targeted depending on the company's resources or needs. In [B2B](http://searchcio.techtarget.com/definition/B2B) marketing, companies are concerned with decision-makers' job titles, the industry sector, whether the company is public or private, its size, location, buying patterns and their technology at their disposal, for example. In [B2C](http://searchcio.techtarget.com/definition/B2C) marketing, companies are concerned with particular customers' profiles, attitudes and lifestyles. Approaches to B2B customer segmentation include vertical or horizontal alignments. In vertical segmentation, companies select certain industries or job titles that would likely find their products appealing and then focus marketing efforts on those segments that they feel are most ready to buy. In horizontal segmentation, companies simply focus on one job title across a wide range of industries and organizations.

**2.4 Digit Recognition:**

Responding to the community's need for better benchmarking, the US National Institute of Standards and Technology (NIST) provided a database of handwritten characters on 2 CD ROMs. NIST organized a competition based on this data in which the training data was known as NIST Special Database 3, and the test data was known as NIST Test Data 1. After the competition was completed, many competitors were distressed to see that although they achieved error rates of less than 1% on validation sets drawn from the training data, their performance on the test data was much worse. NIST disclosed that the training set and the test set were representative of different distributions: the training set consisted of characters written by paid US census workers, while the test set was collected from characters written by uncooperative high school students. Examples from these training and test sets are shown in Figure 1.



Notice that the test images contain some very ambiguous patterns. Although this disparity in distributions is certainly possible in a real world application, it is prudent (and usually possible) to guard against it. In general we can expect best test results when recognizers are tuned to the kind of data they are likely to encounter when deployed. A more subtle, but, for us, a more serious problem arises from having the training and test data belonging to different distributions. Most of our machine learning techniques now use the principles of Structural Risk Minimization 15 in which the capacity (roughly speaking, the number of free parameters) of a classifier is adjusted to match the quantity and the complexity of the training data. Because of the difference in distributions, we cannot use our full machine learning tool set on the NIST data when it is partitioned in this way. For the reasons described above, we repartitioned the NIST data to provide large training and test sets that share the same distribution. We now describe how our new database was created. The original NIST test contains 58,527 digit images written by 500 different writers. In contrast to the training set, where blocks of data from each writer appeared in sequence, the data in the NIST test set is scrambled.



Writer identities for the test set is available and we used this information to unscramble the writers. We then split this NIST test set in two: characters written by the first 250 writers went into our new training set. The remaining 250 writers were placed in our test set. Thus we had two sets with nearly 30,000 examples each. The new training set was completed with enough examples from the old NIST training set, starting at pattern # 0, to make a full set of 60,000 training patterns. Similarly, the new test set was completed with old training examples starting at pattern # 35,000 to make a full set with 60,000 test patterns. In the experiments described here, we only used the first 10,000 test images, but we used the full 60,000 training samples. All the images were size normalized to fit in a 20x20 pixel box (while preserving the aspect ratio). For some experiments, the 20x20 images were deslanted before being presented, i.e. slanted characters were straightened up using moment of inertia methods. For other experiments they were only centered in a larger input field using center of mass. Grayscale pixel values were used to reduce the effects of aliasing. Two methods (LeNet 1 and Tangent Distance) used sub-sampled versions of the images to 16 by 16 pixels. These are the training and test sets used in the benchmarks described in this paper. In this paper, we will call them the MNIST data.

**3.SYSTEM ANALYSIS**

**3.1 Problems with existing system**

* Customers are analyzed only based on their behavior during the trial period of the software
* Clustering algorithms like k-means, GMM, k-median is used to find similarities between the user behavior and the existing data instead of classifying the customer into a specific category.
* Facilities to allow the software provider to select the parameters to analyze the customer are unavailable.
* The approaches to import training data and historic data are complex.
* Lower accuracy
* Different parameters are used for every software being analyzed.

**3.2 Objective**

The Objective is to solve the problem of software providers who have one to many applications that they own and each application has one to many customers using that application. The problem of the software provider is that he/she/organization needs information about the future behavior that the every individual customer might exhibit, instead of just being able to see a list of users

The software provider can use this information to maximize their profit by taking appropriate actions towards specific customers.

The software providers can add and update their applications, view the customers for each application, create their own custom machines to analyze their data be able to add their own training data, identify from past behavior of a customer if the customer may make a purchase next month, identify from past behavior of a customer if the customer is interested in the software

The software provider can save the results of their customer analysis and save it excel sheet. Using the parameters like money spent and access duration the classification can be made on the customer if the customer should be given an offer(discounts) or recommendations for another feature or another software.

**3.3 Motivation**

The Objective is to solve the problem of software providers who have one to many applications that they own and each application has one to many customers using that application. The problem of the software provider is that he/she/organization needs information about the future behavior that the every individual customer might exhibit, instead of just being able to see a list of users

The software provider can use this information to maximize their profit by taking appropriate actions towards specific customers.

**3.4 Proposed System**

* Customers are analyzed based on their past behavior, this behavior can be used to analyze the customer and categorize them based on their behavior.
* Classification algorithms like support vector machines, decision trees and clustering algorithms like k-means, GMM, k-median is used to find similarities between the user behavior and the existing data and classifying the customer into a specific category.
* Facilities to allow the software provider to select the parameters to analyze the customer are available.
* The approaches to import training data and historic data are easy.
* higher accuracy
* Same parameters are used for every software being analyzed.

**3.5 Feasibility Study**

**3.5.1 Programming Language: (C#)**

C# is a simple, modern, general-purpose, object-oriented programming language developed by Microsoft within its .NET initiative led by Anders Hejlsberg. This tutorial will teach you basic C# programming and will also take you through various advanced concepts related to C# programming language.

**3.5.2 IDE: (Visual studios 2015)**

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs for Microsoft Windows, as well as web sites, web apps, web services and mobile apps.

**3.5.3 Machine Learning implementation: (Accord.net)**

The Accord.NET Framework is both a C# machine learning framework and a complete framework for building computer vision, computer audition, signal processing and statistical applications. Sample applications provide a fast start to get up and running quickly, and an extensive documentation helps fill in the details. It is a complete framework for building production-grade computer vision, computer audition, signal processing and statistics applications even for commercial use. A comprehensive set of sample applications provide a fast start to get up and running quickly, and an extensive documentation and wiki helps fill in the details.

**3.5.4 Data Base: (SQL Server 2016)**

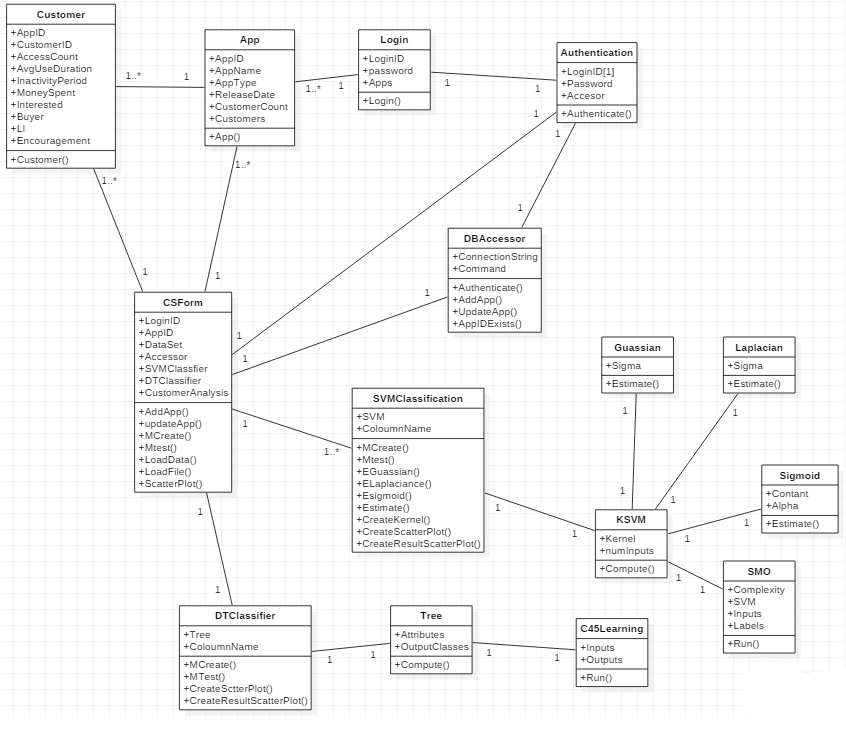
Microsoft SQL Server is a relational database management system developed by Microsoft. As a database server, it is a software product with the primary function of storing and retrieving data as requested by other software applications—which may run either on the same computer or on another computer across a network (including the Internet).

**3.5.5 Data Base connectivity: (ADO.NET)**

ADO.NET is a data access technology from the Microsoft .NET Framework that provides communication between relational and non-relational systems through a common set of components. ADO.NET is a set of computer software components that programmers can use to access data and data services from the database. It is a part of the base class library that is included with the Microsoft .NET Framework. It is commonly used by programmers to access and modify data stored in relational database systems, though it can also access data in non-relational sources.

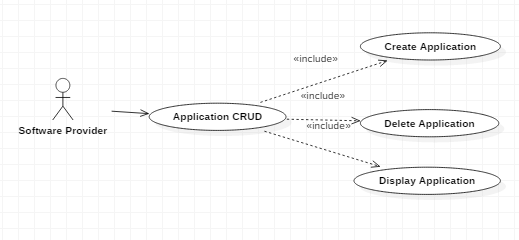
**4.SYSTEM DESIGN**

**4.1 Customer Segmentation Class Diagram:**

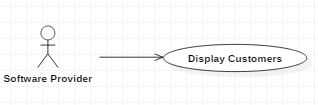
****

**4.2 Use Case Diagram:**

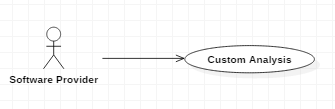
4.2.1 Application CRUD



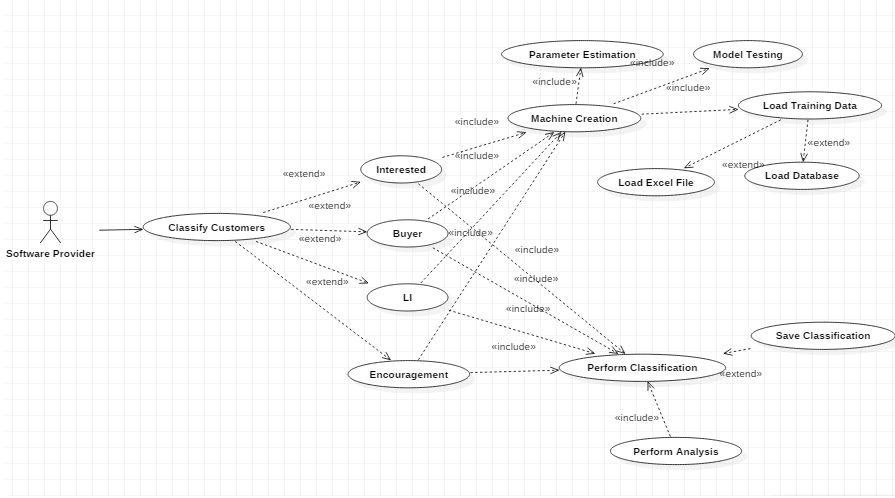
4.2.2 Display Customers:



4.2.3 Customer Analysis:

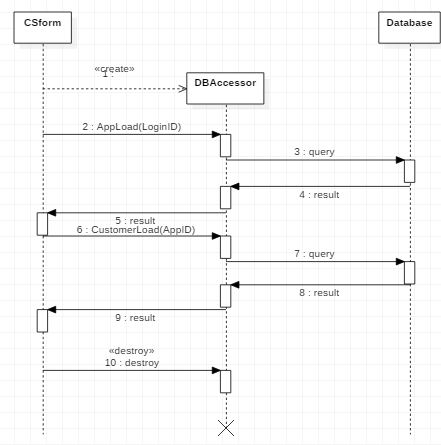
****

4.2.4 Customer Classification:

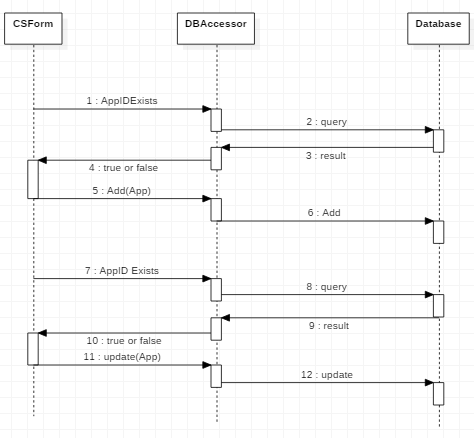


**4.3 Dataflow Diagrams:**

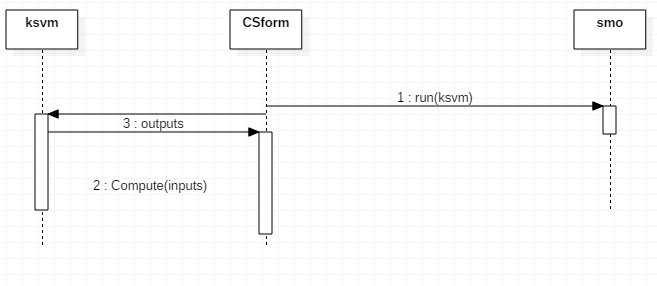
4.3.1 Display Application:

****

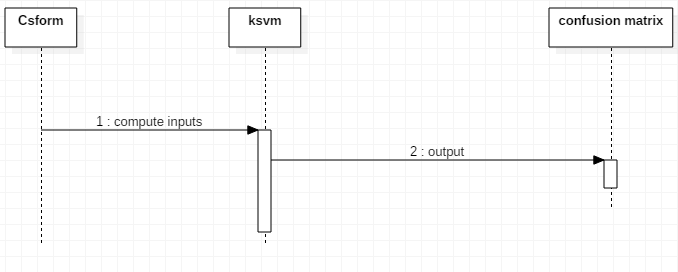
4.3.2 Application Edit:



4.3.3 Classifying Customers:

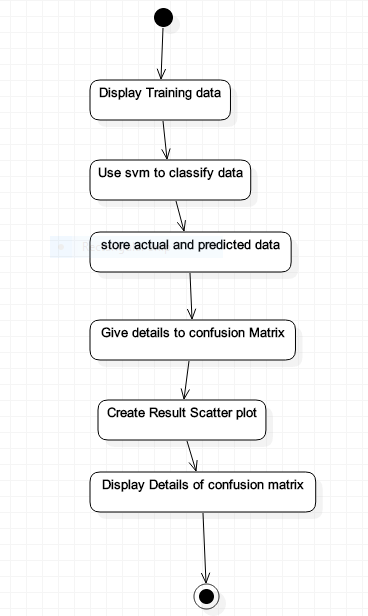


4.3.4 Model Testing:

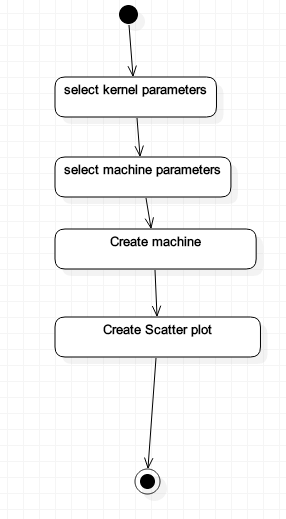


**4.4 Activity Diagram:**

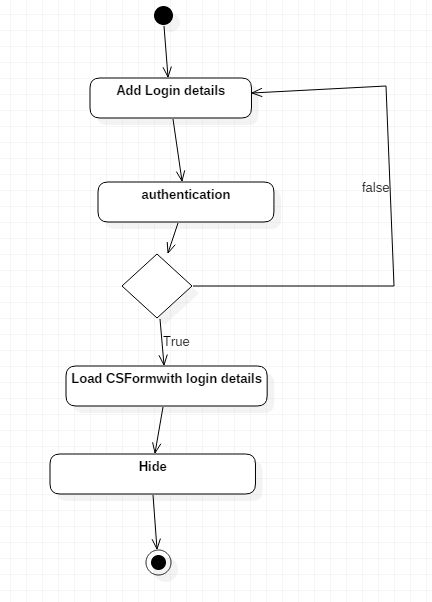
4.4.1 Buyer Machine Creation(Model Testing)

****

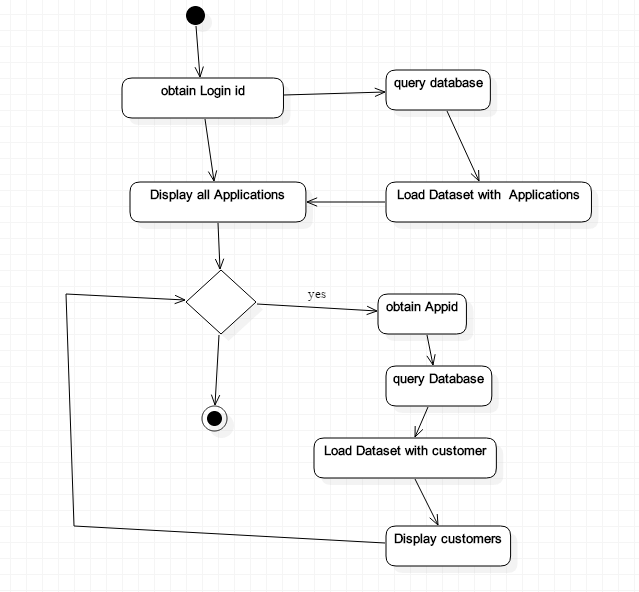
4.4.2 Buyer Machine Creation – Create Machine

****

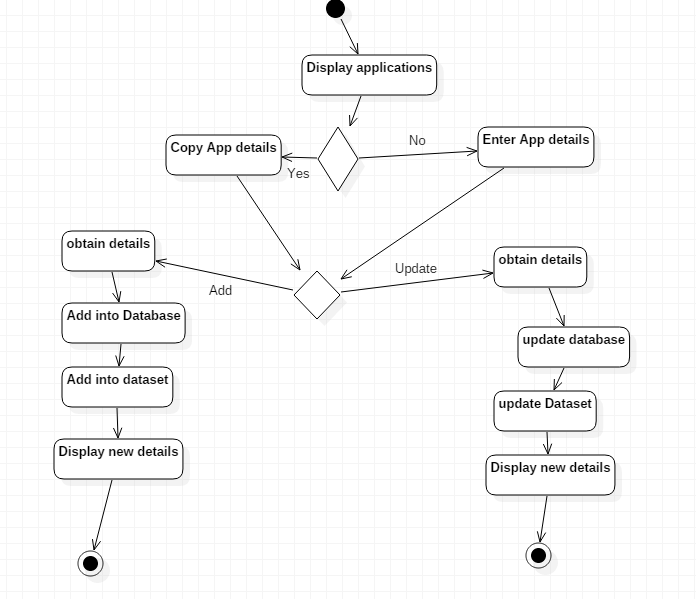
4.4.3 Authentication:



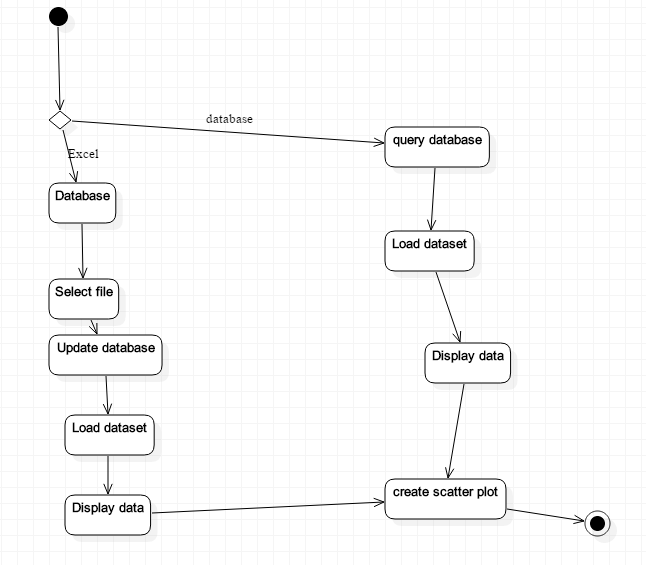
4.4.4 Displaying Application & Customers:



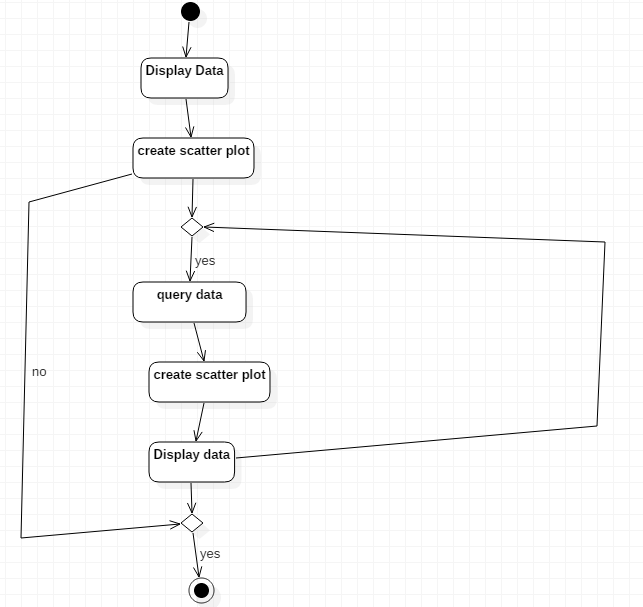
4.4.5 Edit Application:



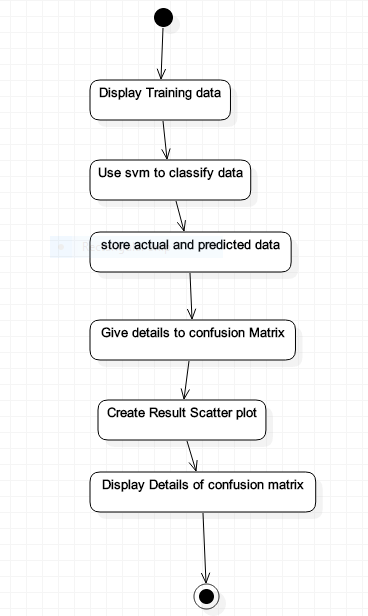
4.4.6 Buyer machine creation - loading training data

****

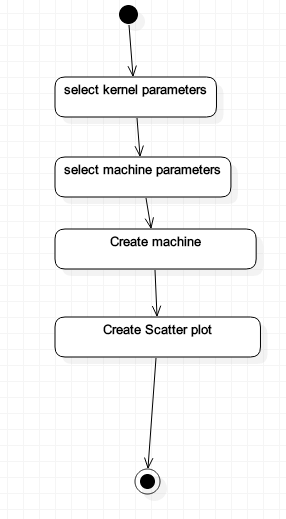
4.4.7 Buyer Customer Analysis:



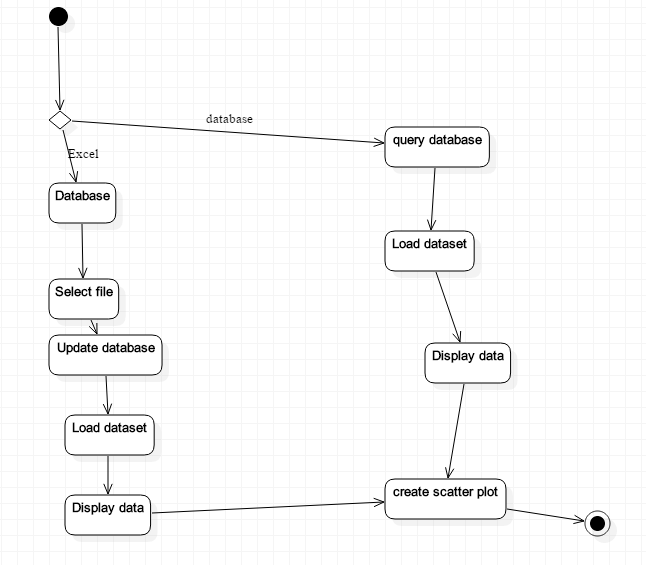
4.4.8 Interested Machine Creation(Model Testing)

****

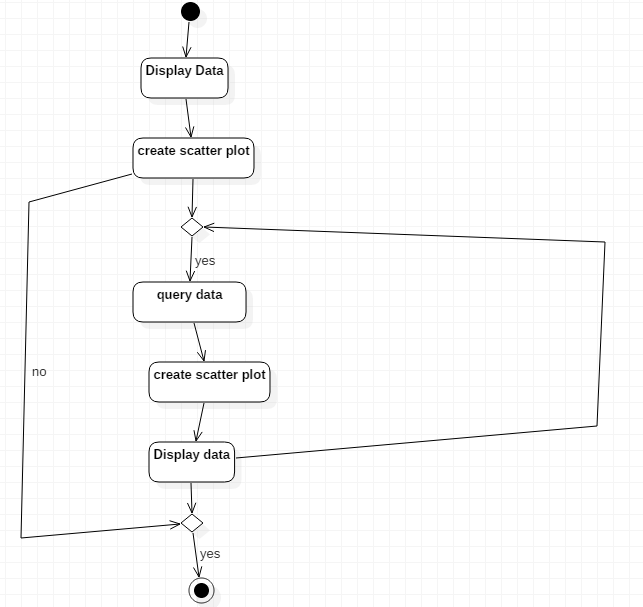
4.4.9 Interested Machine Creation – Create Machine

****

4.4.10 Interested machine creation - loading training data

****

4.4.11 Interested Customer Analysis:



**5.IMPLEMENTATION**

**CUSTOMER SEGMENTATION ANALYSIS FORM:**

namespace SampleApp

{

/// <summary>

/// Classification sample application using Kernel Support Vector Machines.

/// </summary>

///

public partial class MainForm : Form

{

SupportVectorMachine<IKernel> svm;

string[] columnNames; // stores the column names for the loaded data

public MainForm()

{

InitializeComponent();

dgvLearningSource.AutoGenerateColumns = true;

dgvPerformance.AutoGenerateColumns = false;

openFileDialog.InitialDirectory = Path.Combine(Application.StartupPath, "Resources");

}

/// <summary>

/// Creates a Support Vector Machine and teaches it to recognize

/// the previously loaded dataset using the current UI settings.

/// </summary>

///

private void btnCreate\_Click(object sender, EventArgs e)

{

if (dgvLearningSource.DataSource == null)

{

MessageBox.Show("Please load some data first.");

return;

}

// Finishes and save any pending changes to the given data

dgvLearningSource.EndEdit();

// Creates a matrix from the entire source data table

double[,] table = (dgvLearningSource.DataSource as DataTable).ToMatrix(out columnNames);

// Get only the input vector values (first two columns)

double[][] inputs = table.GetColumns(0, 1).ToJagged();

// Get only the output labels (last column)

int[] outputs = table.GetColumn(2).ToInt32();

// Creates a new instance of the SMO learning algorithm

var smo = new SequentialMinimalOptimization<IKernel>()

{

// Set learning parameters

Complexity = (double)numC.Value,

Tolerance = (double)numT.Value,

PositiveWeight = (double)numPositiveWeight.Value,

NegativeWeight = (double)numNegativeWeight.Value,

Kernel = createKernel()

};

try

{

// Run

svm = smo.Learn(inputs, outputs);

lbStatus.Text = "Training complete!";

}

catch (ConvergenceException)

{

lbStatus.Text = "Convergence could not be attained. "+

"The learned machine might still be usable.";

}

createSurface(table);

// Check if we got support vectors

if (svm.SupportVectors == null || svm.SupportVectors.Length == 0)

{

dgvSupportVectors.DataSource = null;

graphSupportVectors.GraphPane.CurveList.Clear();

return;

}

// Show support vectors on the Support Vectors tab page

double[][] supportVectorsWeights = svm.SupportVectors.InsertColumn(svm.Weights);

string[] supportVectorNames = columnNames.RemoveAt(columnNames.Length - 1).Concatenate("Weight");

dgvSupportVectors.DataSource = new ArrayDataView(supportVectorsWeights, supportVectorNames);

// Show the support vector labels on the scatter plot

double[] supportVectorLabels = new double[svm.SupportVectors.Length];

for (int i = 0; i < supportVectorLabels.Length; i++)

{

int j = inputs.Find(sv => sv == svm.SupportVectors[i])[0];

supportVectorLabels[i] = outputs[j];

}

double[][] graph = svm.SupportVectors.InsertColumn(supportVectorLabels);

CreateScatterplot(graphSupportVectors, graph.ToMatrix());

}

private void createSurface(double[,] table)

{

// Get the ranges for each variable (X and Y)

DoubleRange[] ranges = table.GetRange(0);

// Generate a Cartesian coordinate system

double[][] map = Matrix.Cartesian(

Vector.Interval(ranges[0], 0.05),

Vector.Interval(ranges[1], 0.05));

// Classify each point in the Cartesian coordinate system

double[] result = svm.Decide(map).ToMinusOnePlusOne().ToDouble();

double[,] surface = map.ToMatrix().InsertColumn(result);

CreateScatterplot(zedGraphControl2, surface);

}

/// <summary>

/// Tests the previously created machine into a new set of data.

/// </summary>

///

private void btnTestingRun\_Click(object sender, EventArgs e)

{

if (svm == null || dgvTestingSource.DataSource == null)

{

MessageBox.Show("Please create a machine first.");

return;

}

// Creates a matrix from the source data table

double[,] table = (dgvTestingSource.DataSource as DataTable).ToMatrix();

// Extract the first and second columns (X and Y)

double[][] inputs = table.GetColumns(0, 1).ToJagged();

// Extract the expected output labels

bool[] expected = Classes.Decide(table.GetColumn(2));

// Compute the actual machine outputs

bool[] output = svm.Decide(inputs);

// Use confusion matrix to compute some performance metrics

dgvPerformance.DataSource = new [] { new ConfusionMatrix(output, expected) };

// Create performance scatter plot

CreateResultScatterplot(zedGraphControl1, inputs,

expected.ToMinusOnePlusOne().ToDouble(),

output.ToMinusOnePlusOne().ToDouble());

}

private void btnEstimateGaussian\_Click(object sender, EventArgs e)

{

DataTable source = dgvLearningSource.DataSource as DataTable;

// Creates a matrix from the source data table

double[,] sourceMatrix = source.ToMatrix(out columnNames);

// Get only the input vector values (in the first two columns)

double[][] inputs = sourceMatrix.GetColumns(0, 1).ToArray();

DoubleRange range; // valid range will be returned as an out parameter

Gaussian gaussian = Gaussian.Estimate(inputs, inputs.Length, out range);

numSigma.Value = (decimal)gaussian.Sigma;

}

private void btnEstimateLaplacian\_Click(object sender, EventArgs e)

{

DataTable source = dgvLearningSource.DataSource as DataTable;

// Creates a matrix from the source data table

double[,] sourceMatrix = source.ToMatrix(out columnNames);

// Get only the input vector values (in the first two columns)

double[][] inputs = sourceMatrix.GetColumns(0, 1).ToArray();

DoubleRange range; // valid range will be returned as an out parameter

var laplacian = Laplacian.Estimate(inputs, inputs.Length, out range);

numLaplacianSigma.Value = (decimal)laplacian.Sigma;

}

private void btnEstimateSigmoid\_Click(object sender, EventArgs e)

{

DataTable source = dgvLearningSource.DataSource as DataTable;

// Creates a matrix from the source data table

double[,] sourceMatrix = source.ToMatrix(out columnNames);

// Get only the input vector values (in the first two columns)

double[][] inputs = sourceMatrix.GetColumns(0, 1).ToArray();

DoubleRange range; // valid range will be returned as an out parameter

var sigmoid = Sigmoid.Estimate(inputs, inputs.Length, out range);

if (sigmoid.Alpha < (double)Decimal.MaxValue && sigmoid.Alpha > (double)Decimal.MinValue)

numSigAlpha.Value = (decimal)sigmoid.Alpha;

if (sigmoid.Constant < (double)Decimal.MaxValue && sigmoid.Constant > (double)Decimal.MinValue)

numSigB.Value = (decimal)sigmoid.Constant;

}

private void btnEstimateC\_Click(object sender, EventArgs e)

{

DataTable source = dgvLearningSource.DataSource as DataTable;

// Creates a matrix from the source data table

double[,] sourceMatrix = source.ToMatrix(out columnNames);

// Get only the input vector values (in the first two columns)

double[][] inputs = sourceMatrix.GetColumns(0, 1).ToArray();

IKernel kernel = createKernel();

// Estimate a suitable value for SVM's complexity parameter C

double c = kernel.EstimateComplexity(inputs);

numC.Value = (decimal)c;

}

/// <summary>

/// Creates the kernel function specified in the user interface.

/// </summary>

///

private IKernel createKernel()

{

if (rbGaussian.Checked)

return new Gaussian((double)numSigma.Value);

if (rbPolynomial.Checked)

{

if (numDegree.Value == 1)

return new Linear((double)numPolyConstant.Value);

return new Polynomial((int)numDegree.Value, (double)numPolyConstant.Value);

}

if (rbLaplacian.Checked)

return new Laplacian((double)numLaplacianSigma.Value);

if (rbSigmoid.Checked)

return new Sigmoid((double)numSigAlpha.Value, (double)numSigB.Value);

else throw new Exception();

}

private void MenuFileOpen\_Click(object sender, EventArgs e)

{

if (openFileDialog.ShowDialog(this) == DialogResult.OK)

{

string filename = openFileDialog.FileName;

string extension = Path.GetExtension(filename);

if (extension == ".xls" || extension == ".xlsx")

{

ExcelReader db = new ExcelReader(filename, true, false);

TableSelectDialog t = new TableSelectDialog(db.GetWorksheetList());

if (t.ShowDialog(this) == DialogResult.OK)

{

DataTable tableSource = db.GetWorksheet(t.Selection);

double[,] sourceMatrix = tableSource.ToMatrix(out columnNames);

// Detect the kind of problem loaded.

if (sourceMatrix.GetLength(1) == 2)

{

MessageBox.Show("Missing class column.");

}

else

{

this.dgvLearningSource.DataSource = tableSource;

this.dgvTestingSource.DataSource = tableSource.Copy();

CreateScatterplot(graphInput, sourceMatrix);

}

}

}

}

lbStatus.Text = "Switch to the Machine Creation tab to create a learning machine!";

}

#region Scatter plot and Graph creation

public void CreateScatterplot(ZedGraphControl zgc, double[,] graph)

{

GraphPane myPane = zgc.GraphPane;

myPane.CurveList.Clear();

// Set the titles

myPane.Title.IsVisible = false;

myPane.XAxis.Title.Text = columnNames[0];

myPane.YAxis.Title.Text = columnNames[1];

// Classification problem

PointPairList list1 = new PointPairList(); // Z = -1

PointPairList list2 = new PointPairList(); // Z = +1

for (int i = 0; i < graph.GetLength(0); i++)

{

if (graph[i, 2] == -1)

list1.Add(graph[i, 0], graph[i, 1]);

if (graph[i, 2] == 1)

list2.Add(graph[i, 0], graph[i, 1]);

}

// Add the curve

LineItem myCurve = myPane.AddCurve("G1", list1, Color.Blue, SymbolType.Diamond);

myCurve.Line.IsVisible = false;

myCurve.Symbol.Border.IsVisible = false;

myCurve.Symbol.Fill = new Fill(Color.Blue);

myCurve = myPane.AddCurve("G2", list2, Color.Green, SymbolType.Diamond);

myCurve.Line.IsVisible = false;

myCurve.Symbol.Border.IsVisible = false;

myCurve.Symbol.Fill = new Fill(Color.Green);

// Fill the background of the chart rect and pane

//myPane.Chart.Fill = new Fill(Color.White, Color.LightGoldenrodYellow, 45.0f);

//myPane.Fill = new Fill(Color.White, Color.SlateGray, 45.0f);

myPane.Fill = new Fill(Color.WhiteSmoke);

zgc.AxisChange();

zgc.Invalidate();

}

public void CreateResultScatterplot(ZedGraphControl zgc, double[][] inputs, double[] expected, double[] output)

{

GraphPane myPane = zgc.GraphPane;

myPane.CurveList.Clear();

// Set the titles

myPane.Title.IsVisible = false;

myPane.XAxis.Title.Text = columnNames[0];

myPane.YAxis.Title.Text = columnNames[1];

// Classification problem

PointPairList list1 = new PointPairList(); // Z = -1, OK

PointPairList list2 = new PointPairList(); // Z = +1, OK

PointPairList list3 = new PointPairList(); // Z = -1, Error

PointPairList list4 = new PointPairList(); // Z = +1, Error

for (int i = 0; i < output.Length; i++)

{

if (output[i] == -1)

{

if (expected[i] == -1)

list1.Add(inputs[i][0], inputs[i][1]);

if (expected[i] == 1)

list3.Add(inputs[i][0], inputs[i][1]);

}

else

{

if (expected[i] == -1)

list4.Add(inputs[i][0], inputs[i][1]);

if (expected[i] == 1)

list2.Add(inputs[i][0], inputs[i][1]);

}

}

// Add the curve

LineItem

myCurve = myPane.AddCurve("G1 Hits", list1, Color.Blue, SymbolType.Diamond);

myCurve.Line.IsVisible = false;

myCurve.Symbol.Border.IsVisible = false;

myCurve.Symbol.Fill = new Fill(Color.Blue);

myCurve = myPane.AddCurve("G2 Hits", list2, Color.Green, SymbolType.Diamond);

myCurve.Line.IsVisible = false;

myCurve.Symbol.Border.IsVisible = false;

myCurve.Symbol.Fill = new Fill(Color.Green);

myCurve = myPane.AddCurve("G1 Miss", list3, Color.Blue, SymbolType.Plus);

myCurve.Line.IsVisible = false;

myCurve.Symbol.Border.IsVisible = true;

myCurve.Symbol.Fill = new Fill(Color.Blue);

myCurve = myPane.AddCurve("G2 Miss", list4, Color.Green, SymbolType.Plus);

myCurve.Line.IsVisible = false;

myCurve.Symbol.Border.IsVisible = true;

myCurve.Symbol.Fill = new Fill(Color.Green);

// Fill the background of the chart rect and pane

//myPane.Chart.Fill = new Fill(Color.White, Color.LightGoldenrodYellow, 45.0f);

//myPane.Fill = new Fill(Color.White, Color.SlateGray, 45.0f);

myPane.Fill = new Fill(Color.WhiteSmoke);

zgc.AxisChange();

zgc.Invalidate();

}

#endregion

private void toolStripMenuItem5\_Click(object sender, EventArgs e)

{

Close();

}

private void toolStripMenuItem7\_Click(object sender, EventArgs e)

{

new AboutBox().ShowDialog(this);

}

private void numPolyConstant\_ValueChanged(object sender, EventArgs e)

{

}

}

}

**7. Testing**

**1. Login**

|  |  |  |
| --- | --- | --- |
| Test case 1: Login | | Priority : High |
| Test Objective: To log the user into one of the available applications | | |
| Test Description: The user needs to enter the LoginID, password, select the required application and click the enter button | | |
| Requirements Verified: Yes | | |
| Test Environment: A PC | | |
| Test Setup/Pre-Conditions: An Instance of the Authentication form must be created | | |
| **Actions** | **Expected Results** | |
| The user will click the Enter button | An Instance of the selected application is created | |
| Pass: Yes Conditions pass: No Fail: No | | |
| Problems / Issues: NIL | | |
| Notes: Successfully Executed | | |

**2. Display Applications**

|  |  |  |
| --- | --- | --- |
| Test case 1: Display Applications | | Priority : High |
| Test Objective: Display all the softwares belonging to the user that has loged in | | |
| Test Description: When the form loads display all softawares in the datagrid | | |
| Requirements Verified: Yes | | |
| Test Environment: A PC | | |
| Test Setup/Pre-Conditions: An Instance of the CutomerSegmentation form must be created .  Application must have connectivity to the database | | |
| **Actions** | **Expected Results** | |
| When the form loads | All softwares are displayed in the datagrid | |
| Pass: Yes Conditions pass: No Fail: No | | |
| Problems / Issues: NIL | | |
| Notes: Successfully Executed | | |

**3. Display Customers**

|  |  |  |
| --- | --- | --- |
| Test case 1: Display Customers | | Priority: low |
| Test Objective: To Display all the users of the selected Software | | |
| Test Description: The user needs to click an application | | |
| Requirements Verified: Yes | | |
| Test Environment: A PC | | |
| Test Setup/Pre-Conditions: Applications must be loaded in the datagrid.  Database connectivity is required | | |
| **Actions** | **Expected Results** | |
| The user will click an application in the datagrid | Customer of the selected application are displayed in the datagrid | |
| Pass: Yes Conditions pass: No Fail: No | | |
| Problems / Issues: NIL | | |
| Notes: Successfully Executed | | |

**4. Add an Application**

|  |  |  |
| --- | --- | --- |
| Test case 1: Add an Application | | Priority : low |
| Test Objective: Add an Application to the database | | |
| Test Description: when the User enters details of the new application and clicks enter the dataset, database and datagrid is updated and a new appplication is displayed | | |
| Requirements Verified: Yes | | |
| Test Environment: A PC | | |
| Test Setup/Pre-Conditions: An Instance of the CutomerSegmentation form must be created .  Application must have connectivity to the database | | |
| **Actions** | **Expected Results** | |
| The user will click the Add button | New application is displayed in the Datagrid | |
| Pass: Yes Conditions pass: No Fail: No | | |
| Problems / Issues: NIL | | |
| Notes: Successfully Executed | | |

**5. Update an Application**

|  |  |  |
| --- | --- | --- |
| Test case 1: Update an Application | | Priority : low |
| Test Objective: Update an Application in the database | | |
| Test Description: when the user enters details of an application and clicks enter the dataset, database and datagrid is updated and a new details of the application is displayed | | |
| Requirements Verified: Yes | | |
| Test Environment: A PC | | |
| Test Setup/Pre-Conditions : An Instance of the CutomerSegmentation form must be created .  Application must have connectivity to the database | | |
| **Actions** | **Expected Results** | |
| The user will click the update button | New application details is displayed in the Datagrid | |
| Pass: Yes Conditions pass: No Fail: No | | |
| Problems / Issues: NIL | | |
| Notes: Successfully Executed | | |

**6. Create Classifier(Interested)**

|  |  |  |
| --- | --- | --- |
| Test case 1: Create Classifier(Interested) | | Priority : High |
| Test Objective: Creating a new instance of the InterestedClassifer form | | |
| Test Description: User needs to click the Interested menuItem | | |
| Requirements Verified: Yes | | |
| Test Environment: A PC | | |
| Test Setup/Pre-Conditions: An Instance of the CutomerSegmentation form must be created .  Application must have connectivity to the database | | |
| **Actions** | **Expected Results** | |
| The user will click Interested MenuItem | A new instance of the InterestedClassifer form | |
| Pass: Yes Conditions pass: No Fail: No | | |
| Problems / Issues: NIL | | |
| Notes: Successfully Executed | | |

**7. Load training data**

|  |  |  |
| --- | --- | --- |
| Test case 1: Load the training data | | Priority : High |
| Test Objective: The training data should must be retrieved from the database placed in the dataset , selected dimensions must again be loaded into the data set with a different name and it must be displayed on the datagrid | | |
| Test Description: Data is is displayed in the datagrid when theuser clicks load data | | |
| Requirements Verified: Yes | | |
| Test Environment: A PC | | |
| Test Setup/Pre-Conditions: An Instance of the Interested Classifier form must be created .  Application must have connectivity to the database | | |
| **Actions** | **Expected Results** | |
| The user will click the load Data button | Dimensions are displayed in the datagrid | |
| Pass: Yes Conditions pass: No Fail: No | | |
| Problems / Issues: NIL | | |
| Notes: Successfully Executed | | |

**8. Create Machine**

|  |  |  |
| --- | --- | --- |
| Test case 1: Create Machine | | Priority : High |
| Test Objective: Train the decision tree with the training data and display the decision making process in the Treeview and Zedgraph | | |
| Test Description: Decision tree is trained when the user clicks create machine button | | |
| Requirements Verified: Yes | | |
| Test Environment: A PC | | |
| Test Setup/Pre-Conditions: An Instance of the Interested Classifier form must be created . | | |
| **Actions** | **Expected Results** | |
| The user will click the Create Machine button | Decision making Process is shown in the treeview and displayed on he zedgraph | |
| Pass: Yes Conditions pass: No Fail: No | | |
| Problems / Issues: NIL | | |
| Notes: Successfully Executed | | |

**9. Predictive Model Testing**

|  |  |  |
| --- | --- | --- |
| Test case 1: Predictive Model Testing | | Priority: low |
| Test Objective: Testing the performance of the created machine | | |
| Test Description: The Performance of the model is tested and details are displayed on the datagrid | | |
| Requirements Verified: Yes | | |
| Test Environment: A PC | | |
| Test Setup/Pre-Conditions An Instance of the Interested Classifier form must be created . | | |
| **Actions** | **Expected Results** | |
| The user will click the Test Machine button | Details are Displayed on the Datagrid | |
| Pass: Yes Conditions pass: No Fail: No | | |
| Problems / Issues: NIL | | |
| Notes: Successfully Executed | | |

**10. Classifying customers as interested**

|  |  |  |
| --- | --- | --- |
| Test case 1: Classifying customers as interested | | Priority : High |
| Test Objective: Classifying customers of the application as Interested Customers | | |
| Test Description: User needs to click the classify users button | | |
| Requirements Verified: Yes | | |
| Test Environment: A PC | | |
| Test Setup/Pre-Conditions: An Instance of the Interested Classifier form must be created .  Application must have connectivity to the database . | | |
| **Actions** | **Expected Results** | |
| The user will click the Classify Customers button | Customers are classified one by one using the created machine and Interested field of the dataset and database are changed | |
| Pass: Yes Conditions pass: No Fail: No | | |
| Problems / Issues: NIL | | |
| Notes: Successfully Executed | | |

**11. Create Classifier(Buyer)**

|  |  |  |
| --- | --- | --- |
| Test case 1: Create Classifier(Interested) | | Priority : High |
| Test Objective: Creating a new instance of the BuyerClassifer form | | |
| Test Description: User needs to click the Interested menuItem | | |
| Requirements Verified: Yes | | |
| Test Environment: A PC | | |
| Test Setup/Pre-Conditions: An Instance of the CutomerSegmentation form must be created .  Application must have connectivity to the database | | |
| **Actions** | **Expected Results** | |
| The user will click Interested MenuItem | A new instance of the InterestedBuyer form | |
| Pass: Yes Conditions pass: No Fail: No | | |
| Problems / Issues: NIL | | |
| Notes: Successfully Executed | | |

**12. Load training data**

|  |  |  |
| --- | --- | --- |
| Test case 1: Load the training data | | Priority : High |
| Test Objective: The training data should must be retrieved from the database placed in the dataset , selected dimensions must again be loaded into the data set with a different name and it must be displayed on the datagrid | | |
| Test Description: Data is is displayed in the datagrid when theuser clicks load data | | |
| Requirements Verified: Yes | | |
| Test Environment: A PC | | |
| Test Setup/Pre-Conditions: An Instance of the Interested Classifier form must be created .  Application must have connectivity to the database | | |
| **Actions** | **Expected Results** | |
| The user will click the load Data button | Dimensions are displayed in the datagrid | |
| Pass: Yes Conditions pass: No Fail: No | | |
| Problems / Issues: NIL | | |
| Notes: Successfully Executed | | |

**13. Create Machine**

|  |  |  |
| --- | --- | --- |
| Test case 1: Create Machine | | Priority : High |
| Test Objective: Train the decision tree with the training data and display the decision making process in the Treeview and Zedgraph | | |
| Test Description: Decision tree is trained when the user clicks create machine button | | |
| Requirements Verified: Yes | | |
| Test Environment: A PC | | |
| Test Setup/Pre-Conditions: An Instance of the Interested Classifier form must be created . | | |
| **Actions** | **Expected Results** | |
| The user will click the Create Machine button | Decision making Process is shown in the treeview and displayed on he zedgraph | |
| Pass: Yes Conditions pass: No Fail: No | | |
| Problems / Issues: NIL | | |
| Notes: Successfully Executed | | |

**14. Predictive Model Testing**

|  |  |  |
| --- | --- | --- |
| Test case 1: Predictive Model Testing | | Priority: low |
| Test Objective: Testing the performance of the created machine | | |
| Test Description: The Performance of the model is tested and details are displayed on the datagrid | | |
| Requirements Verified: Yes | | |
| Test Environment: A PC | | |
| Test Setup/Pre-Conditions An Instance of the Interested Classifier form must be created . | | |
| **Actions** | **Expected Results** | |
| The user will click the Test Machine button | Details are Displayed on the Datagrid | |
| Pass: Yes Conditions pass: No Fail: No | | |
| Problems / Issues: NIL | | |
| Notes: Successfully Executed | | |

**15. Claasifying customers as interested**

|  |  |  |
| --- | --- | --- |
| Test case 1: Classifying customers as interested | | Priority : High |
| Test Objective: Classifying customers of the application as Interested Customers | | |
| Test Description: User needs to click the classify users button | | |
| Requirements Verified: Yes | | |
| Test Environment: A PC | | |
| Test Setup/Pre-Conditions: An Instance of the Interested Classifier form must be created .  Application must have connectivity to the database . | | |
| **Actions** | **Expected Results** | |
| The user will click the Classify Customers button | Customers are classified one by one using the created machine and Interested field of the dataset and database are changed | |
| Pass: Yes Conditions pass: No Fail: No | | |
| Problems / Issues: NIL | | |
| Notes: Successfully Executed | | |

**6. Output Screenshots**

**6.1 Authentication**

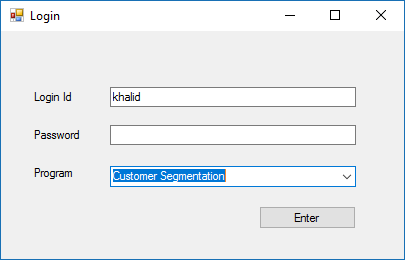
****

Fig: 6.1

**6.2 Displaying Applications:**

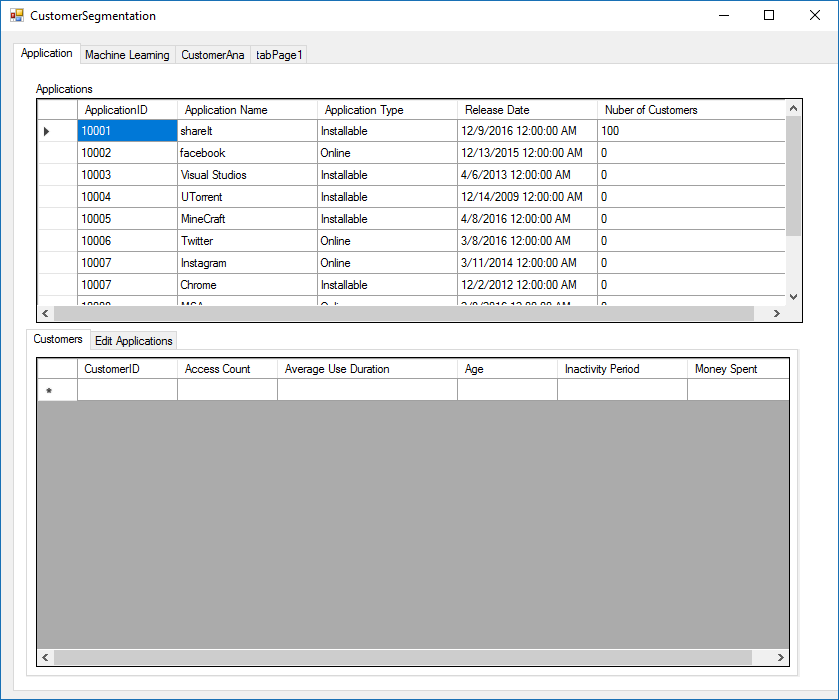
****

Fig: 6.2

**6.3 Displaying Customers:**

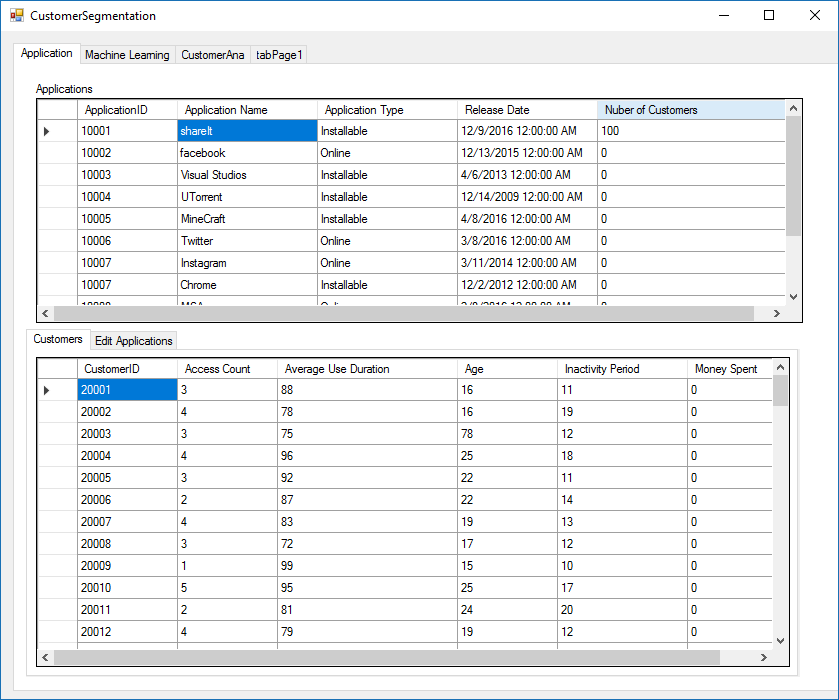
****

Fig: 6.3

**6.4 Editing Applications:**

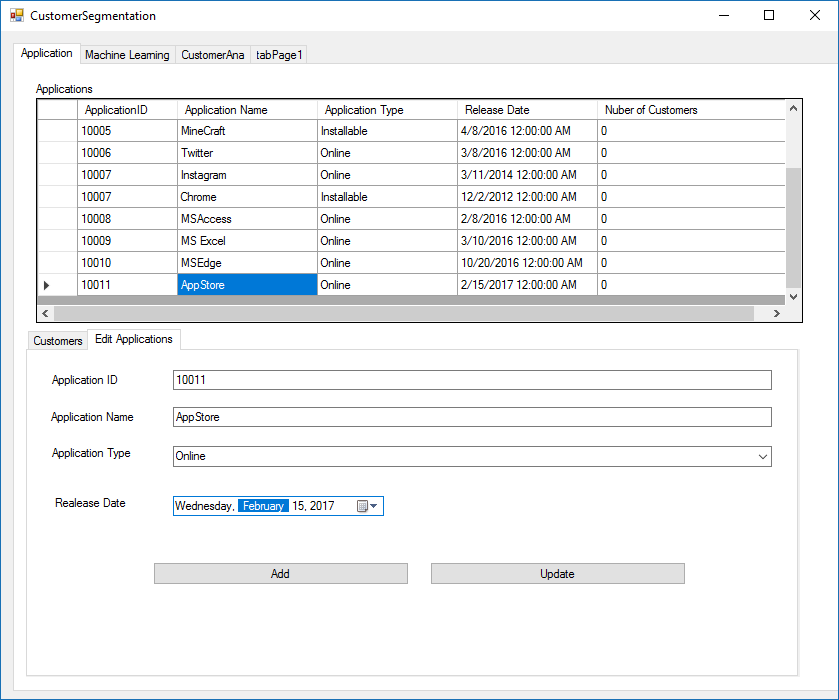
****

Fig: 6.4

**6.5 Loading Training Data:**

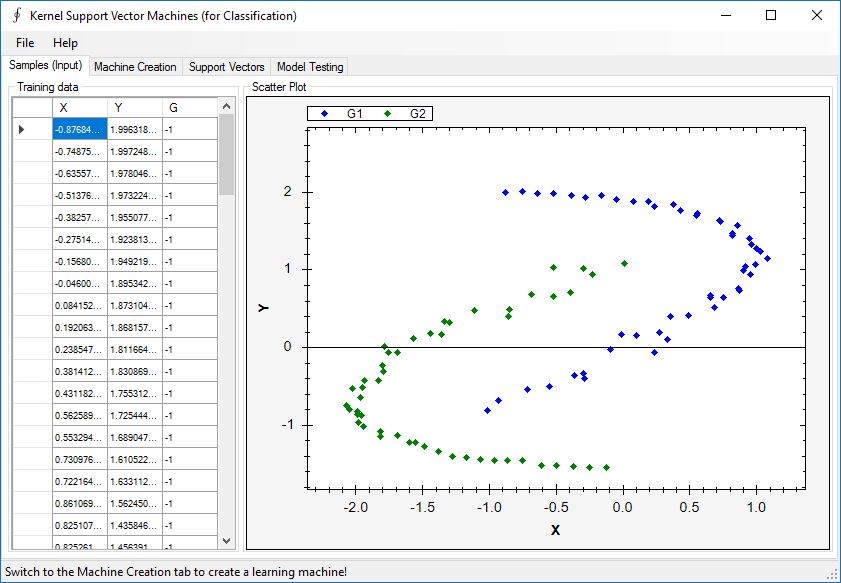
****

Fig: 6.5

**6.6 Creating Machine:**

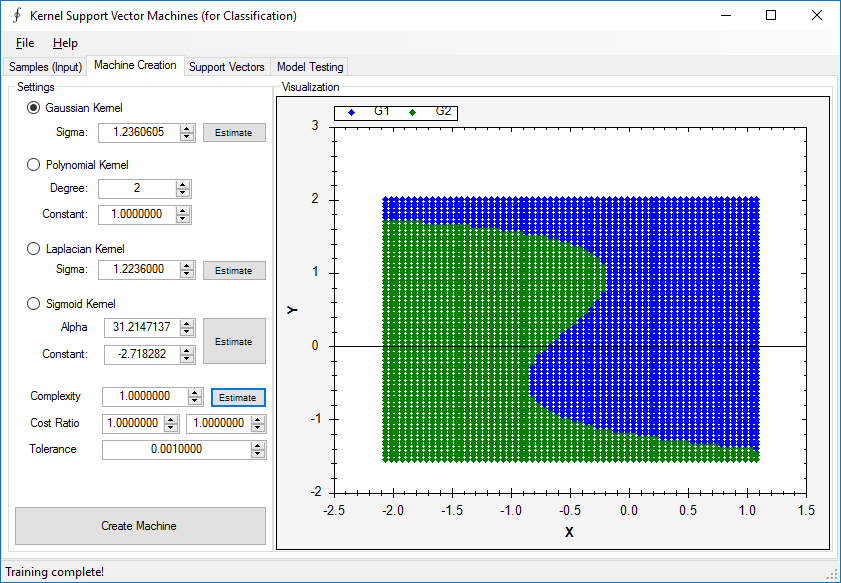
****

Fig: 6.6

**6.7 Testing the Predictive Model:**

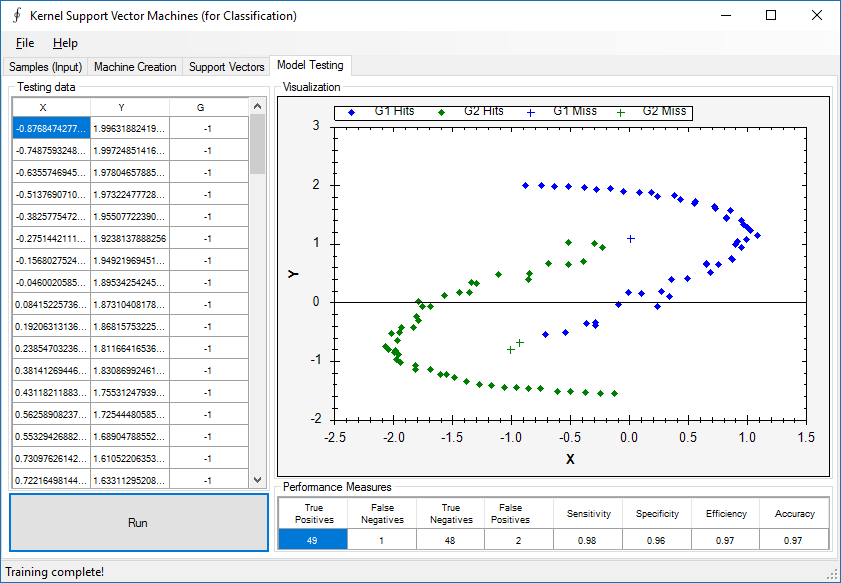


Fig: 6.7

**7.CONCLUSION**

**8.BIBLOGRAPHY & REFRENCES**

a. www.machinelearningmastery.com

b. Yann LeCun, “LEARNING ALGORITHMS FOR CLASSIFICATION: A COMPARISON ON HANDWRITTEN DIGIT RECOGNITION”

c. www.Wikipedia.com

d.www.accord-framework.net