

OPTIMIZED PREDICTION MODEL OF DIABETIC PATIENTS WITH HIDDEN MARKOV MODELS

A DISSERTATION
SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF

**MASTER OF TECHNOLOGY
IN
COMPUTER SCIENCE & ENGINEERING**

Submitted by:
VAIBHAV GAUTAM
2K16 / CSE / 18

Under Supervision of:
Ms DIVYASHIKHA SETHIA
(Assistant Professor)



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY**

(Formerly Delhi College of Engineering)
(Bawana Road, Delhi-110042)

JULY, 2018

DECLARATION

I, **Vaibhav Gautam**, 2K16/CSE/18 student of M.Tech (Computer Science & Engineering), hereby declare that the project dissertation titled “**Optimized Prediction Model of Diabetic Patients With Hidden Markov Models**” which is submitted by me to the Department of Computer Science & Engineering, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any degree, Diploma Associateship, Fellowship or other similar title or recognition.

Place: Delhi

Date:

Vaibhav Gautam
(2K16/CSE/18)

CERTIFICATE

I hereby certify that the Dissertation titled “**Optimized Prediction Model of Diabetic Patients With Hidden Markov Models**” which is submitted by **Vaibhav Gautam**, 2K16/CSE/18, Computer Science & Engineering, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of degree of Master of Technology, is a record of the project work carried out by the student under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi

Date:

Ms Divyashikha Sethia
(Assistant Professor)

ABSTRACT

The extraordinary advances in biotechnology and wellbeing sciences have prompted a unique creation of information, for example, high throughput genetic information and clinical data, produced from expansive Electronic Health Records (EHRs). To this end, utilization of machine learning and information mining techniques in biosciences are direct, like never before previously, essential and critical in endeavors to change brilliantly all accessible data into valuable information.

In this research work, we have used an all-female dataset [19] which comprises eight attributes, i.e., Pregnancies, Glucose, Blood Pressure, Skin Thickness, Insulin, BMI, Diabetes Pedigree Function, Age and two class labels, i.e., 1 or 0 which indicates if a person has diabetes or not.

Firstly, we used supervised techniques on the given dataset and tested their performance, i.e., accuracy in the prediction of people having diabetes. The results for various supervised algorithms are: K-NN - 64.91%, SVM - 65.91%, Random Forest - 67.80%, Gaussian Naive Bayesian - 64.36%, Decision Tree - 63.86%, Logistic Regression - 60.71%. From these results, we conclude that these algorithms cannot predict with sufficient accuracy which stands to be at least 80%.

The unique feature about the supervised technique is if these algorithms encounter a specific attribute which dominates our prediction models, these algorithms always comes out to be on top. The result above means that our dataset has many features essential for prediction. As a result, the supervised Hidden Markov Model as an alternate technique for prediction.

In our prediction methodology, with the help of the Hidden Markov Model, we use Viterbi and BestFirst Search algorithms as decoding techniques. These decoding techniques come up with following subsequences after the Forward-Backward algorithm gives emission probabilities. These probabilities belong to different states of our data model. As a result of which learning by our model gets completed. We have split the dataset into two parts 90% of the dataset as a learning data and 10% of the dataset as the testing data. The accuracy of our model was found out to be 81.81% with Viterbi and 79.22% with BestFirst. Hence, supervised HMM performs better as compared to the supervised techniques used generally.

Important Keywords: Artificial Intelligence, Machine Learning, Diabetes, Viterbi, Best-First Search, Hidden Markov Models, Forward-Backward Algorithm, e.t.c

ACKNOWLEDGEMENT

While bringing out this dissertation to its final form, I came across some people whose contributions have been helpful in carrying out this project. I would like to express my deep and sincere gratitude to our Project Guide, Assistant Professor Ms. **Divyashikha Sethia** for giving me this opportunity and providing invaluable guidance throughout this Project. Her dynamism, vision, sincerity, and motivation have deeply inspired me. It was a great privilege and honor to work and study under her careful supervision and guidance. Her involvement with originality has triggered and nourished the intellectual maturity that helped me for a long time to come.

I express my thankfulness to our Head of Department **Dr. Rajni Jindal** for her continuous support and providing well-equipped labs and other research facilities.

I would like to extend my thanks to Lab Staff at DTU for their continuous support and guidance. My appreciations extend to all our colleagues and friends who have helped us out by sharing their valuable knowledge and views.

I am highly thankful to my parents for their support throughout this project. They have been a source of inspiration, without which completing this project would have been a difficult task.

Contents

1	INTRODUCTION	1
1.1	Overview	1
1.2	Motivation	2
1.3	Problem Statement	3
2	Literature Review	4
2.1	Machine Learning	4
2.2	Decision Trees	5
2.3	Bayes Theorem	6
2.4	Naive Bayes	7
2.5	K-Nearest Neighbour	8
2.6	Random Forest	9
2.7	Logistic Regression	10
2.8	Support Vector Machines	11
2.9	Hidden Markov Models	13
2.10	Viterbi	14
2.11	BestFirst Search	15
2.12	Forward Backward	16
2.13	Software Libraries	17
2.13.1	Pandas	17
2.13.2	Matplotlib	17
2.13.3	NumPy	17
2.13.4	SciKit Learn	18
2.13.5	Seaborn	18
2.13.6	Seqlearn	19
2.14	Performance Metrics	19
2.14.1	Confusion Matrix	19
2.14.2	Basic Terms	19
3	Related Work	21
3.1	Hidden Markov Model	21
3.2	Prediction models related to Diabetes	21
3.3	Hybrid Models OF HMM	22
4	Methodology	23
4.0.1	Feature extraction	23
4.0.2	Training	24
4.0.3	Prediction	24

5	Results	25
6	Conclusions	30
7	Future Scope	31
	Bibliography	32
	Appendix A Code for Data Analyses	34
	Appendix B Code For Supervised Techniques	37
	Appendix C Code For HMM Library	40
	Appendix D Main Code	42

List of Figures

2.1	Decision Tree	6
2.2	K-NN with three neighbors and two labels	8
2.3	Random Forest	10
2.4	Logistic Regression	11
2.5	Hyperplane Selection Example	12
2.6	Hidden Markov Model	13
4.1	HMM with Viterbi	23
4.2	HMM Function	24
5.1	Percentage vs. Number of Pregnancy Histogram	25
5.2	Correlation Matrix Between Attributes	26
5.3	Gini index	27
5.4	Supervised Techniques Performance Histogram	28
5.5	HMM using Viterbi	29
5.6	HMM using Best First	29

List of Tables

5.1	Correlation table	26
5.2	Supervised Results	28

Chapter 1

INTRODUCTION

1.1 Overview

Diabetes is at our doorstep and is ready to become another epidemic in the world and add sorrows to people. If not diagnosed at earliest possible could cause an individual to undergo many problems like hypertension, cardiovascular diseases and dimming of the eyesight. We should be careful with our balanced diet if we are suffering from it. In this dissertation, we see a scientific view towards this disease and try to make a machine learning model which could help many people by predicting based on specific features.

In chapter 1, we discuss the introduction of diabetes, types of diabetes and consensus about it and how it has affected the world. Most important is to understand our problem and what it hints towards our achievements.

In chapter 2, we have studied, the literature content necessary to achieve our goals. It includes the study of various supervised and unsupervised techniques. We have also included the software libraries which i have used to implement the supervised HMM-based model for our project. We discuss some algorithms related to dynamic sub-sequencing like the Viterbi algorithm.

In chapter 3, we discuss the various research materials we have gone through to understand diabetes as a disease, and it is features which are common in predicting it. This material was helpful in our research. It also helped us give a clear understanding of the disease and problem associating with it.

In chapter 4, we have talked about our implemented design of our machine learning model, which is supervised HMM.

In chapter 5, we have our results about the accuracy with HMM and details about the dataset.

In chapter 6, we have the comparison with the base paper we have considered and conclude our other findings. [9].

In chapter 7, we have discussed the prospects of the model and how could improvements be made in the accuracy and learning technique.

1.2 Motivation

Diabetes is one of the major public issues in the world which is rapidly becoming an epidemic. In a report mentioned more than 246 million people are affected by diabetes out of which 46% are in the age group of 40 - 59 this means that younger people are more affected by it. This number is estimated to go up to by 380 million by 2025. It must be baffling to know that about 18 million people who died of cardiovascular diseases have diabetes and hypertension as the predisposing factors in it. The much-growing concern is that a big chunk of people who are having diabetes is people in developing which constitutes of 80% of affected people in the world. Diabetes is of three types Type-1, Type-2, and Gestational diabetes. Type-1 relates to the issues when the pancreas does not create insulin to control blood sugar, whereas in type-2 body respond to the insulin formed within the human body. Gestational diabetes always happens to the pregnant women. It is not very dangerous in most of the cases but might lead to type-2 diabetes.

Seeing the above scenario, if we can predict if the person has diabetes or not on the following attributes Pregnancies, Glucose, Blood Pressure, Skin Thickness, Insulin, BMI, Diabetes Pedigree Function, Age e.t.c. It would turn out to be excellent as the people take notice before it becomes a fatal problem for the future to other organs which gets affected by it like eyesight weakening.

Well, there have been publications where they have predicted with the help of supervised learning techniques to identify if a person possesses diabetes or not. Now the datasets used by them have static attributes which are necessary for supervised methods to predict correctly and have been pretty successful. These again highlights the fact that when such attributes are present, it is easy for these algorithms but when such attributes are absent it is difficult for these algorithms to predict efficiently. Hence that is why we worked with supervised HMM.

1.3 Problem Statement

In our project study, first of all, we have to employ many classifier based prediction techniques to know how well they predict. Secondly, try to implement prediction with the help of the Hidden Markov Model which uses decoder Viterbi and BestFirst Search.

In this work, we use a supervised Hidden Markov Model for prediction modeling and compare its performance against supervised techniques for a specific dataset. The supervised technique prediction makes a one-off decision making which means that they need a static feature to build a model work with accuracy.

We bring out a comparative study between Gill et al. [9] and our model so we can say that how our model performs to the one mentioned in [9].

Chapter 2

Literature Review

2.1 Machine Learning

Arthur Samuel coined the term Machine Learning in the year 1959 [18]. It is a subset of artificial intelligence which often uses statistical techniques which gifts the computer the ability to learn with data. Well to do this we do not have to explicitly programme as it can learn and make predictions from data by exploring the study and construction from the algorithms. These algorithms have overcome the instruction of the static program by making data-driven predictions or decisions. Machine learning algorithms employed with many different types of tasks where programming and designing are almost infeasible because such algorithms which perform with reasonable accuracy is difficult. Machine learning makes predictions with the help of computers by focusing on the computational statistics.

Machine Learning algorithms, widely classified into two categories are-

1. Supervised Learning:

It is one of the machine learning technique where learning function, based on the input-output pairs known as the training examples, should be able to map an input to the output. These algorithms consider the information from labeled training data which consists of a set of training examples(data). In these learning techniques, every example is a pair of consisting inputs objects and mapping to the desired output values. The supervised learning algorithm produces an inferred function after analyzing the training data and which got used for the test data. It would be a great scenario that it can predict the correct class for unseen instances. There are different types of supervised techniques are:-

- Semi-Supervised Learning:

In this, the computers fed a dataset of missing values.

- Active Learning:

In this, the computers can learn about the labels for a limited set of instances. Therefore, they have to optimize their choice to acquire the labels.

- Reinforced Learning:

In this, the data fed as feedback to the processes takes action in a dynamic environment.

2. Unsupervised Learning:

These machine learning algorithms have a task to infer a function that describes an unclassified data's structure. Since the data fed as training examples to the algorithms are unlabelled, it is a gruesome task to compute the accuracy of a structure which is provided

by the algorithm.

We have used many supervised learning algorithms on our datasets:-

2.2 Decision Trees

It is one of the widely used algorithms in the field of data science and also belonging to the class of supervised algorithms. Decision tree algorithm is used to solve problems of classification and regression. Decision Tree creates a training model to predict class or value of targeted variables which it learns by the decision rules from the inferred dataset. Decision tree uses a tree-like representation in which each leaf node corresponds to the class labels and an internal node corresponding to the attribute.

Pseudocode:-

- At the root of the tree, the best attribute of the dataset positioned.
- The training set splits into subsets. The formed subsets are to be such that they contain a similar trait incentive.
- Until all the leaf nodes of all the branches found, It is mandatory to repeat the above steps on each subset.

This implementation of Decision Tree is used to identify which attribute is to consider at the root at each level and node. Handling it could be called an attribute selection.

1. Information Gain – In this, we try to estimate the information contained by each attribute. To measure the randomness or uncertainty of a random variable(X) is defined by entropy (Ex).

The formula used for entropy:-

$$H(X) \doteq EX(I(x)) \doteq 1 * \sum p(x) * \log(p(x)), \text{ for all } x \in X$$

p(x): probability

$$\text{Information Gain} = E(\text{Target}) - E(\text{Target} | D)$$

2. Gini Index – It is a measure of how a randomly chosen variable would incorrectly identify. Attribute with lower Gini index is always preferred. It is defined as:

$$\text{GiniIndex} = 1 - \sum (p_j)^2$$

Overfitting Problem

It is a practical problem that appears while building the decision tree model. It appears when the algorithm goes deeper and deeper to reduce the training set error but results in increased test Error (accuracy goes down). It generally happens when it builds many branches due to outliers moreover, irregularities in data.

Two approaches to tackle this problem are:-

- Pre-Pruning – In this approach tree construction halts in early stages which will if the measure is below the threshold.
- Post-Pruning – In this approach, it goes deeper into the construction of tree, but when it finds overfitting problem, then it runs pruning..

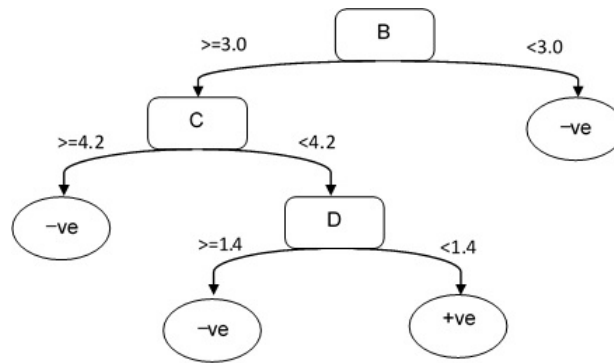


Figure 2.1: Decision Tree

In figure- 2.1, it shows how the decision tree expands based on the information gain values it has and goes further until it reaches a threshold value or no attributes are left to explore.

We have used the CART algorithm as logic for our decision trees and for the trees we have set default values for the depth of the tree which is five, minimum sample split is two, and the minimum leaf samples is one.

Advantages

- Decision Tree is easy to explain & understand.
- It follows the same approach as humans generally follow while making decisions.
- Visualizations simplify Interpretation of a Decision tree model.
- Some hyperparameters that are tuned to be almost null.

Disadvantages

- There is a high probability of occurrence overfitting in the decision tree.
- Generally, it gives low prediction accuracy of datasets as compared to other machine learning
- Calculations become complex when there are many class labels.

Applications

1. Text Processing - A variant of Decision Tree ID3 used for medical text classification.
2. Physics - It is used to identify physical particles.
3. Pharmacology - Tree-based classification used for analysis of drugs.
4. Object Recognition - Used for 3-d objects

2.3 Bayes Theorem

This theorem got discovered by the scientist Rev. Thomas Bayes [13]. This theorem works on the concept of Conditional Probability. Conditional Probability means that something has a probability of happening, something has already occurred is given. We can calculate the probability of an event by using the conditional probabilities, and an event with its prior knowledge.

$$P(H|E) = P(E/H) * P(H)/P(E)$$

Where

$P(H)$: Probability of hypothesis being True, also known as prior probability

$P(E)$: Probability of evidence

$P(E|H)$: Probability of evidence when hypothesis is true

$P(H|E)$: Probability of hypothesis when evidence is true

2.4 Naive Bayes

It is a classifier that gives good results when applied to the textual data analysis, widely used for Natural Language Processing. It works on the principle of Bayes Theorem. This algorithm is straightforward & powerful for classification tasks. It is an excellent technique to use when the dataset contains millions of data tuples with fewer attributes to obtain results. Naive Bayes computes probability in such a manner that the data or record as belonging to a particular class can be found out by comparing the probabilities. The class which has the maximum probability is considered to be the class of the tested data or record; It is called Maximum A Posteriori (MAP).

MAP(H)

$$= MAP(P(H|E))$$

$$= MAP((P(E/H) * P(H))/P(E))$$

$$= MAP(P(E/H) * P(H))$$

We cannot remove P(E) because removing it will not affect the results but helps to normalize the data and is called evidence probability. This classifier makes assumptions that attributes are unrelated and it does not affect any attribute that some of the attributes got removed. In datasets, we have to test our hypothesis against given multiple pieces of evidence or features.

$$P(H|MultipleEvidences) = (P(E1/H)*P(E2/H)*...*P(En|H))/P(MultipleEvidences)$$

Types of Naïve Bayes are:-

- Gaussian Naïve Bayes- This type of Naive Bayes, used when attribute values are continuous then it makes assumptions that every class distribution is gaussian according to values associated with it.

$$P(x_i|y) \doteq (1/(\sqrt{2\pi}(\sigma_y)^2)) * (-(x_i - \mu_y)^2/2 * \sigma_y^2)$$

- MultiNomial Naïve Bayes- It is preferred to use this technique when we have our data multinomially distributed. It applies in text categorization.
- Bernoulli Naïve Bayes- This Naive Bayes when we have the multivariate Bernoulli distribution which means that we have multiple attributes which are binary valued.

Advantages

- These algorithms are highly fast and scalable.
- It is of use in the classification of multiclass and binary data
- It is used to train on the small dataset.

Disadvantages.

- It cannot learn the relationship between features as it assumes them unrelated

Applications

- Sex Classification - to classify a given person is male or female in a big dataset.
- Document Classification - It is a problem to identify documents with their contents.

2.5 K-Nearest Neighbour

Fix, and Hodges [13] first proposed K-NN classifier in the year 1951 for performing pattern classification task. It is a type of supervised classifier. This technique uses the information of neighbor points to make predictions about an object's target class. The simplest version of this algorithm is to predict the class by finding the class which is nearest. The closest class is identified using the distance metrics like Euclidean distance.

Pseudocode:-

1. Compute the distance between the points such as " $d(x, x_i)$ " $i=1,2,\dots,n$; where d stands for distance.
2. Sort in non-decreasing order all the n euclidean distances.
3. Let m be the positive integer and take the first m -distances from the sorted list.
4. Corresponding to m -distances, we have to search all the m -points.
5. Let m_i denotes the number of points belonging to the i^{th} class among m -points, i.e., $m_i \geq 0$.
6. If $m_i > m_j$ & $i \neq j$ then place x in the target label class i .

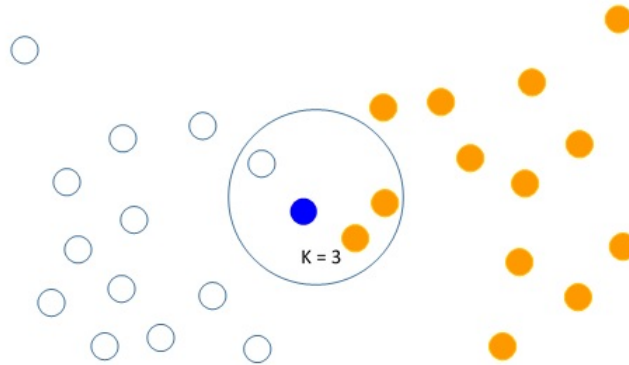


Figure 2.2: K-NN with three neighbors and two labels

In figure 2.2, we can see that we need to find the class label for the blue ball and to determine if we have defined the value of k as 3 and we can see by the mechanism of voting the blue ball belongs to that of light maroon balls and not the white balls..

In K-NN, K is the number of neighbors selecting which in itself is a critical problem. A low k value means that noise has a higher influence on the result which means that overfitting problem becomes high. A very high k -value mean that it becomes very costly and time taken by the algorithm is very high. We have used a ball tree algorithm for our K- Nearest Neighbour implementation with the number of neighbors at default to be five.

Advantages

- Implementations are simple.
- Quick executions did for small datasets.

- Performance is similar to that of the Bayes classifier.
- Any prior knowledge regarding the dataset is not required.

Disadvantages

- Time increases to preprocess a large dataset.
- Distance computations, As a result, much time is taken to come up with results for the larger dataset.

Applications

- Concept Search - It is used to find the documents with a similar concept.
- Closest Pair Problem - It is a problem to identify the closest pair.

2.6 Random Forest

Random Forest algorithm used for both kinds of problems classification and regression. It is one of the supervised algorithm techniques. As the name suggests, it makes a forest with some trees. More the number of trees, denser will be the forest and hence as the population of trees increases, more the accuracy and gives a higher accuracy of results.

It uses the concepts of decision tree algorithm but doesn't use the standard computation of information gain or the Gini index approach for decision making. Pseudocode for this algorithm split is into two stages:-

- Random forest creation.
- Prediction approach.

Random Forest Creation

1. Selection of "n" features from "t" attributes at random where $n < t$.
2. Among the "n" features we need to calculate the node "d" using best-split point strategy.
3. To get the best-split point, we need to split the nodes into daughter nodes.
4. Repeat all the above steps till the algorithm reaches "T" number of nodes.

Prediction Approach

1. Use the rules of every randomly created decision tree to get the outcome and store it. So the test features could be tested against it.
2. Votes of each predicted target need to calculate.
3. The result of the final prediction rests on the highest voted predicted target as the final predicted class by the random forest.

We have used ten estimators in our random forest classifier with the Gini index as the criterion; minimum sample split as two and leaf samples as one.

Advantages

- It is in use for both classification and regression tasks.
- It handles missing values in the datasets.

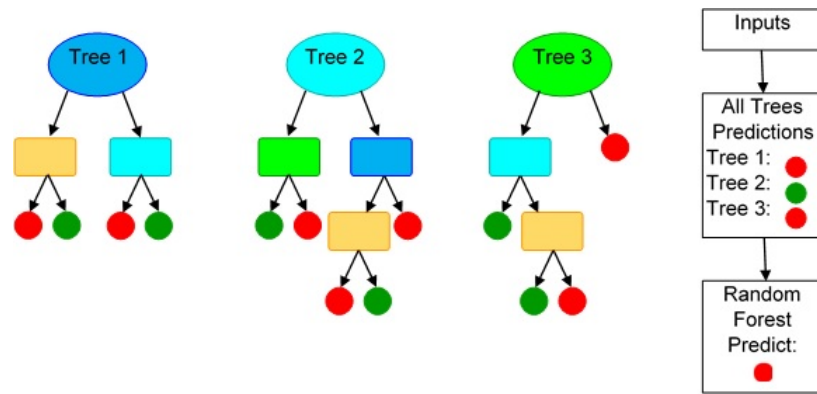


Figure 2.3: Random Forest

In figure 2.3, we can see that the structure made by the decision tree for the different trees of the forest and in the flow diagram above we can see that the tree-1 and tree-3 are predicting the same class label so as a result, the predicted class label is a red color.

- When we have some trees, the classifier will not overfit model.
- The classifier can also model for categorical values.

Applications

- Banking - Used for finding loyal and fraud customers
- Medicine - Used for finding correct combinations of drugs to validate medicine.
- Stock Market - Used for identifying stock behavior and finding the expected profit or loss for buying stocks.
- E-commerce - Used in recommendation engines.

2.7 Logistic Regression

It is a classifier which uses the calculated logits (Score) to predict the targeted class. It measures the relationship between the categorical dependent variable and one or more independent variable by estimating probabilities using a logistic function.

Formula for logit calculation is

$$\ln(1 - p/p) = \beta_0 + \beta_1 * X_1 + \dots + \beta_k * X_k$$

All X are variables and the β 's are the weights

Types of logistic Regression models

1. Binary Logistic Regression- It has only two possible outcomes true or false.
2. Multinomial Logistic Regression- It has more than two possible outcomes or categories.
3. Ordinal Logistic Regression - It has three or more categories with orderings.

Decision Boundary

A threshold taken, to predict which class a data belongs. Based upon this threshold, the obtained estimated probability is classified. Decision boundary can be linear or non-linear. The polynomial order can be increased to get a complex decision boundary.

Cost Function

The formula for cost function is

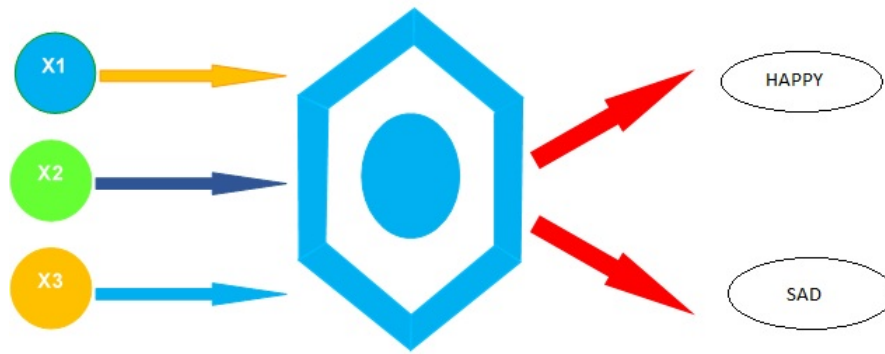


Figure 2.4: Logistic Regression

In figure 2.4, the X_1 , X_2 , X_3 are inputs while the θ_1 , θ_2 , θ_3 are the weights and the output to be calculated is happy or sad.

$$\cos(h\theta(x), Y(actual)) \doteq \log(h\theta(x)) \text{ if } Y = 0 \text{ or } \log(1 - h\theta(x)) \text{ if } Y = 1$$

Advantages

- It may handle nonlinear effects.
- In variance assumption, there is no homogeneity.
- Error terms which are normally distributed are not assumed.

Applications

- Image Segmentation and Categorization - This technique is used in the segmentation of an image in different and uses to categorize feature so that it can identify it.
- Geographic Image Processing - This technique is also used Geographical image processing by the scientists to find out information about the land and process such information.
- Handwriting recognition - It is also used to identify the handwriting of the individual.

2.8 Support Vector Machines

In 1992, Vapnik, Guyon & Boser invented Support Vector Machines [18]. It is said to be one of the most dominant classification algorithms. It used as both a supervised technique as well as unsupervised technique. SVM can also work if it does not have features or class labels.

There are two kinds of SVM classifiers:-

1. Linear SVM classifier
2. Non Linear SVM classifier

Linear SVM classifier

In this classifier model, it expects that the data points to be separated by a gap. It predicts by dividing two classes with the help of hyperplane. The primary focus for drawing a hyperplane is to maximize the distance between either datapoint of class and hyperplane. The resulted hyperplane called as the maximum-margin hyperplane.

Non-Linear SVM Classifier

Our dataset is generally dispersing, and as a result, it gets difficult to choose a clear hyper-plane boundary. To solve this nonlinear problem, the suggestion of this classifier came out. In this, a kernel trick applied by the maximum-margin hyperplanes. Data points plotted in higher dimensional space. Some Kernels are:-

1. polynomial kernel

- Homogeneous Kernel:- $k(\vec{x}_i, \vec{x}_j) \doteq (\vec{x}_i \cdot \vec{x}_j)$
- Non Homogeneous Kernel:- $K(x, y) \doteq (x^t y + c)^d$

2. Radial Basis Function Kernel:- $K(x, x') \doteq \exp(-(\|x - x'\|^2)/2\sigma^2)$

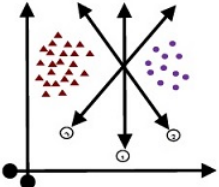
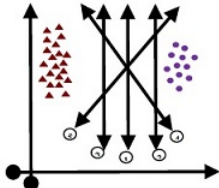
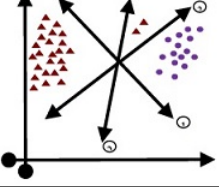
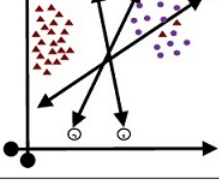
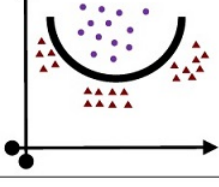
Figure	Explanation
	1 st decision boundary chosen as it is maximizing between both the class levels.
	1 st decision boundary shows the maximum distance from hyperplanes of both levels.
	3 rd boundary is separating all triangles and circle class levels so it will be chosen.
	1 st decision boundary is the chosen one as it shows maximum boundary
	We have to use a different kernel as the like radial basis for further classification.

Figure 2.5: Hyperplane Selection Example

In figure 2.5, we have two classes Δ & O . Their are hyperplanes given and in the explanation which hyperplane is to choose is provided by maximizing hyperplane.

We have used support vector classification with the help of libsvm support library(integrated support software for classification for support vector). The kernel used is the radial basis function. It is also called C support vector classification.

Advantages

- It is useful when the number of features is substantial.
- It works doesn't get affected if the number of features is more than samples.
- With the kernel, hyperplanes built for the use of non-linear data classification.
- It is a robust model to solve prediction problems since it maximizes margin.

Disadvantages

- Choice of the kernel is difficult. If chosen incorrectly could lead an increase in error rate.
- As the samples increases, it gives poor performance.
- It has high algorithmic complexity and also needs a prodigious memory.
- Generalizations could be very slow in the test phase.

Applications

- It finds application in facial recognition.
- It finds application in speech recognition.
- Information Retrieval for text characterization.

2.9 Hidden Markov Models

Hidden Markov Model is a statistical variant of Markov model [18] where the system which gets modeled is considered to be in Markov process with hidden states. We can easily represent these HMM models as Bayesian Networks.

In Markov models, the observer can directly view the states, and therefore the parameter they have is the state transition probabilities, but states are not directly visible in the hidden Markovian models. In HMM, the output based on the states which are visible. Each & every state has a probability distribution over the possible output sequences. Well, the tokens or sequences generated by the Hidden Markov models reveals some knowledge about these sequence of states. An HMM can be seen as speculation of blend that demonstrate where the shrouded factors (or inert factors), which control the blended segment chosen for every perception, are joined through a Markov procedure instead of autonomous of each other. As of late, concealed Markovian models have added up to Markovian pairwise models and Markovian triplet models which permit thought of magnificent unpredictable structures of information and the demonstration of non-stationary information.

The figure- 2.6 depicts the normal process of an HMM. In the figure- 2.6, we can see oval

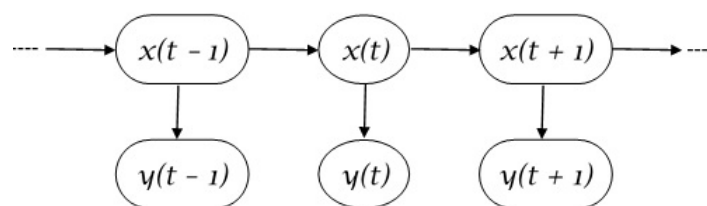


Figure 2.6: Hidden Markov Model

shapes which represent a random variable that can take any value. $y(t)$, random variable, at time t is known as the observation. The $x(t)$, random variable, at time t is the hidden state. The arrows symbolize the dependencies among various states. It is clear from figure-2.6 that distribution of conditional probability $x(t)$, depends only on the $x(t-1)$ value. The values before $(t-1)$ has no impact on the corresponding state at time t . So also, the estimations of the observed variable $y(t)$ rely upon the estimation of shrouded variable $x(t)$ both in the meantime.

Applications

- System State identification.
- Characterizing human gait.
- Biological Sequence representation identification.

2.10 Viterbi

Viterbi is a dynamic programming algorithm. Andrew J. Viterbi found this algorithm in 1967 [18] and hence, named after him. This algorithm is used to find the most likely sequence of hidden states known as the Viterbi path which results in the sequence of observed events.

Input:-

- Representation of state space is given as $S \doteq \{ S_1, S_2, \dots, S_N \}$.
- Representation of observation space is given as $O \doteq \{ O_1, O_2, \dots, O_K \}$.
- $N \times N$ is the transition matrix size of A , matrix A_{ij} stores all the probability which gives knowledge about transition from state s_i to s_j state.
- $N \times k$ is the size of emission matrix B , matrix B_{ij} contains all the probabilities of observation o_j from state s_i .
- N is the size of the array which contains initial probabilities π , $\pi[i]$ contains the probability of state s_i at time $t=1$.
- Recorded sequence of observations in the algorithm will be $\{ y_1, y_2, \dots, y_T \}$.

Output:-

The hidden state sequence as the most likely output is given as $X \doteq \{ x_1, x_2, \dots, x_T \}$.

Algorithm

in the algorithm k & s belongs 1 to N

viterbi(O, S, π, A, T, B): X

- for each state s from 1 to N do
 - $vit[s, 1] \leftarrow \pi_s * B_{s, o_1}$
 - $backpointer[s, 1] \leftarrow 0$
 - for each time step t from 2 to T do
 - * $vit[s, t] \leftarrow \max_k vit[k, t-1] * A_{k, s} * B_{s, o_t}$
 - * $backpointer[s, t] \leftarrow \text{argmax}_k (vit[k, t-1] * A_{k, s} * B_{s, o_t})$

```

        * end for
    - end for

    •  $Z_T \leftarrow \text{argmax}(vit[s, T])$ 

    •  $X_T \leftarrow S_{Z_T}$ 

    - for  $i \leftarrow T, T-1, \dots, 2$  do
        *  $Z_{i-1} \leftarrow \text{backpointer}[Z_i, i]$ 
        *  $X_{i-1} \leftarrow S_{Z_{i-1}}$ 
    - end for

    • return X

```

end function

The time complexity for viterbi is $O(T * N^2)$

Applications

The Viterbi algorithm finds application in these fields

- Dial-Up Modems - This algorithm used in our modems for signaling.
- Deep-space Communications - This applies to satellite communications.
- Speech Recognition - It also applies to speech recognition of a person.
- Natural Language processing - It is applied in understanding the text written by a person.

2.11 BestFirst Search

In both the algorithms of Breadth-First Search and Depth-FirstSearch, to choose next node we choose the adjacent node in these algorithms. Both BFS and DFS blindly explore the paths in the graphs. These algorithms do not have the heuristic function to guide them in the state space search to find the next most promising node. It is the heuristic type search algorithm.

In this algorithm, we use a priority queue so that we can store the cost of exploring the nodes in it. It is the slight variation of BFS so that we use the priority queue instead of the queue.

Algorithm

B-F-S(Graph g, Node Start)

1. Create a new priority queue as p .
2. Insert Start as the first node in priority queue p.
3. Until p is empty
 - $v = p.\text{DeleteMin}$ - it delete the node with minimum cost
 - if (v is found to be goal)
 - program exit and print the path.
 - else
 - for each neighbour u of v

- If u is unmarked or unvisited
- mark u as visited or marked
- $p.insert(u)$ - It inserts u in the priority queue p
- mark u as examined
- end for

4. End of B-F-S

2.12 Forward Backward

The forward-backward algorithm is in use as an inference application related to the Hidden Markov Models as they compute all the hidden state variables, posterior marginals, the sequences of emission $O_{1:T} \doteq O_1, \dots, O_T$, i.e., it calculates the probabilities of all hidden states, i.e., $P(X_t|O_{1:T})$. This task for inference is called smoothing. This algorithm uses a dynamic paradigm approach to calculate the posterior marginal distributions in into two passes..

Both passes go opposite each other in a direction following their names in the model to calculate the probabilities of states. To sum up the above algorithm we can say:-

1. calculates the probabilities in the forward direction.
2. calculates the probabilities in the backward direction.
3. calculates the values which help in smoothening of our results.

When algorithms that operate on the sequential model in two loops like forwarding & backward manner like the forward-backward algorithm. Then they belong to the same class of algorithms.

2.13 Software Libraries

We have used Python to implement our project, and it offers many libraries to complete our work.

2.13.1 Pandas

Pandas is an open source library, under BSD license. It is a library which provides high performance. It has data structures easy to use and tools used for the Python. Python as a language is not very good for data analysis and preparation, as it is for munging and preparation of data. Pandas library helps to fill this gap and enables us in carrying out data analysis workflow in Python as well.

This library has specific highlights:-

- It contains a DataFrame object which is so efficient that it can quickly do manipulations of data & help in indexing.
- For reading & writing data, we have in-memory data structures which can also work with different formats like CSV & excel.
- Flexible reshaping and pivoting in datasets.
- It has intelligent data alignment which can handle the missing data.
- High performance of merging and joining of datasets.
- Highly optimized for performance
- Python with pandas is used in various places having academic and commercial domains.

2.13.2 Matplotlib

It is a 2D plotting library for Python. It creates quality figures in many formats and interactive environments. Matplotlib tries to make things a lot easier & possible. We can generate plots like error charts, power spectra, bar charts, histograms, scatterplots, e.t.c, with just a few lines of code.

Pyplot in matplotlib module which provides a MATLAB-like interface. It is designed to be in use with MATLAB, with the ability to use Python, and also gives us the advantage of being a free and open source.

Several toolkits are available extends MATLAB:-

- Basemap: Used for plotting map's, coastlines and geographical boundaries.
- Excel tools: It used in for the purpose to exchange of data with Microsoft Excel.
- Mplot3d: It helps in making 3-D plots
- Natgrid: If interfaced with this library could easily able to grid space data.

2.13.3 NumPy

Scientific computation with Python is the most basic capability of NumPy. It contains other things like:

- It has an N-dimensional array object
- It has broadcasting functions
- It can easily integrate with programming languages like c/c++ and Fortran code
- It is in use for linear algebra, Fourier transform, & random number.

Besides its regular scientific uses, NumPy is also used to store generic data efficiently which has multi-dimensional traits. Data-types can be defined arbitrarily. It helps NumPy to integrate with different databases speedily and with ease.

2.13.4 SciKit Learn

It is an open source library, commercially usable and is under BSD license. It extends NumPy, SciPy, and matplotlib. It has efficient tools for analysis of data & mining. It is accessible for everyone and reused in various contexts.

NumPy used for various applications like

- Classification - It helps to identify which category an object is belonging.
- Regression - It helps in predicting a continuous-valued attribute of the object.
- Clustering - It helps in the grouping of similar objects in a set like arranging.
- Dimensionality Reduction - To reduce many random variables.
- Model Selection – It compares, validates and choose parameters for model selection.
- Pre-processing - Feature extraction and normalization.

2.13.5 Seaborn

Seaborn is a matplotlib based visualization library. To draw attractive graphics, it provides high-level interfacing with other libraries. Features this library possess are:-

Features this library has are:-

- It has many built-in themes which help in styling graphics from matplotlib.
- It has tools which choose palettes of different colors to make which reveals the pattern in our data.
- It has visualizing functions for multivariate distributions and can easily compare them.
- It has tool functionality for different kinds of dependent and independent variables so that they can fit and visualize linear regression models.
- It has functionality, to discover structure it can use a clustering algorithm and visualize matrix data
- It has various methods which can quickly plot the time-series data with efficient estimation.
- To build complex visualizations, we need high-level abstractions for structuring grid plots which are present in this library.

Intentions for making this library was to make understanding & exploring the data as a chief part. Seaborn's, plotting functions work on the dataframes & arrays, which can take the whole dataset as input. This library performs all the operations of statistical model fitting & aggregation internally to produce plots which are illustrious & informative. It makes the whole lot of things easy for any developer.

The plotting capacities attempt to accomplish something helpful when called with a negligible arrangement of contentions, and they uncover various adaptable choices through extra parameters. A bit of the ability plot particularly into a matplotlib tomahawks question, as others take a shot at an entire figure and try to make plots with a couple of boards. In these last cases, the plot is drawn using a Network dissent that associations the structure of the figure to the dataset.

2.13.6 Seqlearn

Seqlearn inherits the properties of sci-kit learn machine library so that it can handle sequence classification: observation sequence must be labeled individually, but the order in which they appear must be kept intact.

Seqlearn imitates the fit/predict application program interface of the sci-kit learn library. Seqlearn is compatible with data formats of sci-kit learn library. It adds an extra argument to the sci-kit learn method which can encode the input structure. It is known as length; it should be of an array of integers which denotes the length of sequences in (X,y).

Dataset Utilities

It uses various dataset utilities:

- It loads CONLL type files which are used to extract features and vectorize them.
- It used for mapping symbolic input features to columns.

Model Selection & Evaluation

It uses a sequence aware k-fold CV splitter which partitions the entered sequences into many set-fragments with an equal number of samples while the sequence is kept intact. To do this, they use a greedy approach.

Sequence Classifier

- It uses first-order HMM to with multinomial event model. This multinomial model uses both Viterbi and best first for sequencing [12].

2.14 Performance Metrics

2.14.1 Confusion Matrix

A confusion matrix is a diagrammatic representation which helps to determine how the classification model has performed on a set of test data for which the results are already known. It is easy to understand, but the terminology related to this is itself very confusing.

2.14.2 Basic Terms

- **True Positives (TP):** These are the type of cases, where our prediction states, yes, meaning person have the disease, and in actuality, they do suffer from the disease.
- **True Negatives (TN):** Our prediction says no, meaning the person does not have the disease, and in reality, they are not suffering from the disease.
- **False Positives (FP):** Our prediction says yes, meaning the person has the disease, and but they are not suffering from the disease. It is called “Type 1 error”.
- **False Negatives (FN):** Our prediction says no, but they do have the disease. It is called “Type 2 error”.

List of rates computed from confusion matrix are:-

- **Accuracy** – This tells us that how much classifier is correct. It is computed by $((TP+TN)/Total)$.
- **Misclassification rate** – It tells that often it is wrong. It is $(1-Accuracy)$. It is computed by $((FP+FN)/Total)$. It is also known as “Error Rate”.
- **True Positive Rate** – It tells how often it predicts yes. It is also called “Sensitivity” or “Recall”.
- **False Positive Rate** –It tells how often predicts yes when it is no. It is computed by $(FP/Actual\ No)$.
- **Specificity** –When it does it no when it is no. It is computed by $(TN/Actual\ No)$. It is called $(1- False\ Positive\ Rate)$.
- **Precision** – When it predicts yes and how often it is correct. It is computed by $(TP/Predicted\ yes)$.
- **Prevalence** - It tells how often does the yes situation occur in our sample. It is computed by $(Actual\ Yes / Total)$.

Here. We have a confusion matrix for a binary classifier where prediction is either a yes or no as we have taken into account of 200 patients with a disease.

N=200	Predicted YES	Predicted NO
Actual Yes=150	TP=135	FN=15
Actual No=50	FP=15	TN=35

Accuracy – 85%, Misclassification rate – 15%, True Positive Rate – 90%, False Positive Rate – 30%, Specificity – 70% , Precision – 85%, Prevalence – 75%.

Chapter 3

Related Work

3.1 Hidden Markov Model

Bengio et al. [14] work related to HMM has already started taking place in speech signals, where the authors have proposed an integrated model where they have used the ANN with HMM to model the speech signals and have been successful. They have used ANN for training and integration with the HMM model and was available depict essential acoustic parameters of speech signals. As a result, the HMM integrated with ANN was used to develop speech recognition system. Marton in [15] has used HMM in the field of study of phenomenography which yields to the fact how HMM used in applications of thinking and learning. They have made results by categorizations of descriptions and the categorizations which are more critical. Seymore et al. [16] have used HMM in information extraction, primarily to understand the structure of the model learned from the data. They have used their data to extract the essential headings in research papers related to computer science with the accuracy of 92.9%.

3.2 Prediction models related to Diabetes

Honeycutt et al. [4] in their work they have forecasted that how many people in the US have diabetes by 2050 have proposed a dynamic Markov model which helps in forecasting by age, race, ethnicity, and sex. In this model, they have identified people under three categories, i.e., diagnosed and not diagnosed with diabetes and death. Tennvall et al. [5] have predicted how they can prevent the cost of foot ulcers and amputations with the help of Markov Model simulations. Kuo et al. [2] have used markovian model predictions concerning the survivability rate of people having asymptotic and asymptotic type-2 Mellitus and found people with asymptotic type-2 mellitus have more survival rate

Cho et al. [5] they have used irregular and unbalanced data to predict diabetic nephropathy using the visualization and feature selection methods as they have the various features, so they have used SVM for prediction results approx.92%. Priya et al. [7] have used Support vector machines for predicting diabetic retinopathy that how diabetes affects our retina and accuracy was 98.92%. Tapak et al. [8] have used supervised techniques for the prediction of diabetes in Iranian people have found SVM as the better to predict diabetes with the accuracy of 89%. Sundaram et al. [11] using recurrent neural networks in diabetes prediction and is the best neural network variant as the accuracy was found to be approx 95%. Rejeski et al. [17] they have used HMM to characterize disability states of a person within the risk group created and to estimate

the functional decline used mixed-effect ordinal logistic regression. Weight loss and improved fitness resulted in the decline of mobility in overweight adults with type-2 diabetes.

3.3 Hybrid Models OF HMM

Gill et al. [9] have made a hybrid mix of supervised and unsupervised techniques for the prediction of people who have diabetes. In this, they have used an all-female diabetic dataset having attributes Glucose, Pregnancies, Blood Pressure, Skin Thickness, Insulin, BMI, Age, DiabetesPedigree Function, and Outcome. For the attribute selection, they have used gain ratio. First of all, they have used the following supervised techniques SVM - 77.47%, K-NN - 69.79%, Naive Bayes - 76.30%, Decision Tree - 74.21%, and Neural Networks - 75.13%. They have used HMM as clustering techniques after finding associations of features with outcome using fuzzy logic. For training, they have used ANN techniques, and after this, they have fed ANN output to the HMM. As a result, the proposed model has an accuracy of 92% with HMM. They have used fuzzy with ANN as a training method as a forward feed to the HMM which makes it complicated and costly. Statistical attributes found by them is glucose, insulin, and body mass which according to us are not sufficient attributes for prediction. They have used many data mining tools to come up with the fuzzy logic decision and able to complete a supervised cum unsupervised hybrid model. Their model is an hybrid and able to come up with statistical information, but we cannot improve our results with the same inferred decisions.

Chapter 4

Methodology

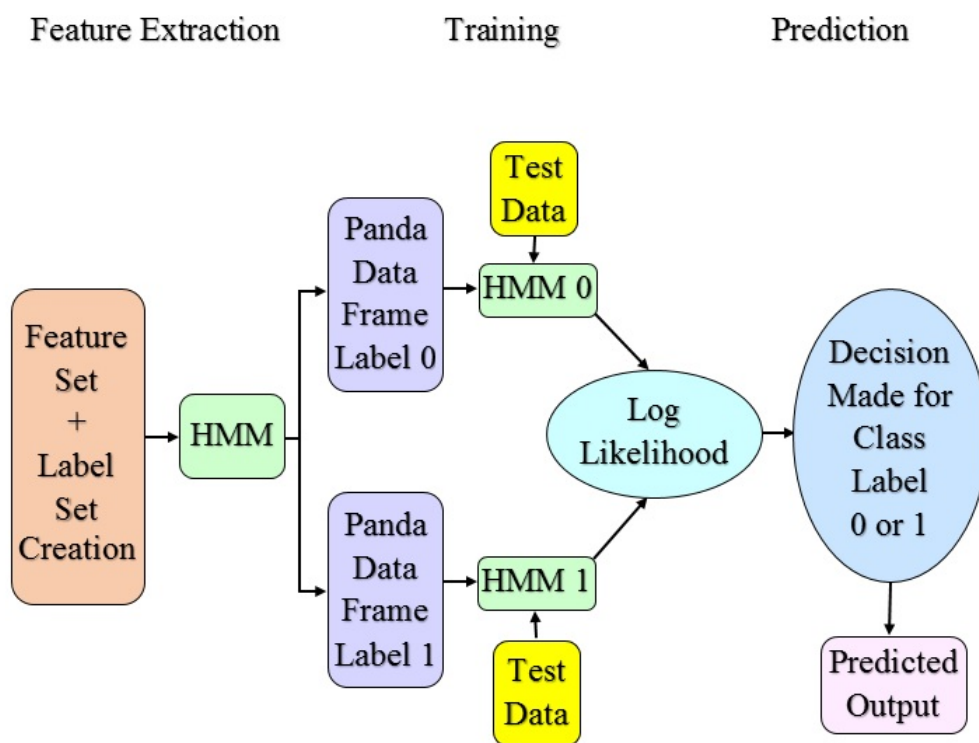


Figure 4.1: HMM with Viterbi

Every machine learning model has three phases:-

1. Feature Extraction
2. Training
3. Prediction

4.0.1 Feature extraction

Here in this first stage, we extract class labels and the various attributes in our model so that can understand the data. After understanding the data, the output in panda frames moves to the training phase of the model.

4.0.2 Training

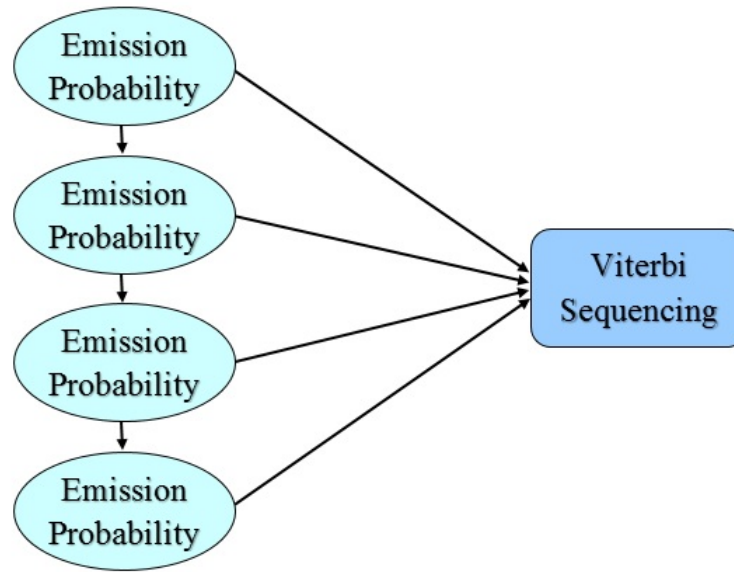


Figure 4.2: HMM Function

We feed our result from the first stage, i.e., panda data frames to our general HMM which is perceptron based and compute the emission probabilities by applying the Forward-Backward algorithm. After that, we use Viterbi(Best First)sequencing and arrange the data in panda dataset frames which arrange them in two panda dataset frames which label 0 or 1 and then feed it into two respective HMM trained units which are specific for their class labels and test data brought in these HMM trained units. Both the train HMM0 and HMM1 will compute the log-likelihood function which will tell to which class label will it belong to 0 or 1. Suppose the if a particular person test data goes through these models and HMM0 computes less than 0.5 and HMM1 will compute more than 0.5 so the person belongs to class label one which means he has diabetes.

4.0.3 Prediction

In predicted values, we compare with the actual outcome of the test data and try to come up with different performance metrics that how the model has performed. After the predicted output result is given it compares with actual data and confusion matrix shows on the screen.

Chapter 5

Results

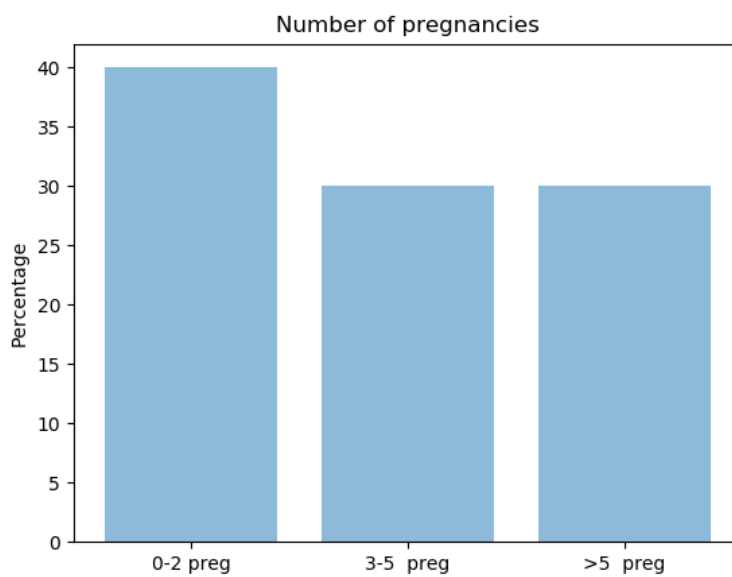


Figure 5.1: Percentage vs. Number of Pregnancy Histogram

In the figure 5.1, we can see that approximately 40% of women have pregnancies in the range of 0-2 and about 30% each for the group having pregnancies in range of 3-5 & >5. We find that women who are in the pregnancy region of above 2 are more prone to have diabetes.

Table 5.1: Correlation table

	Pregnancy	Glucose	BP	SkinThick.	Insulin	BMI	DPF	Age	Outcome
Pregnancy	1.000	0.129	0.141	-0.081	-0.073	0.017	-0.033	0.544	0.221
Glucose	0.129	1.000	0.152	0.057	0.331	0.221	0.137	0.263	0.466
BP	0.141	0.152	1.000	0.207	0.088	0.281	0.041	0.239	0.065
SkinThick.	-0.081	0.057	0.207	1.000	0.436	0.392	0.183	-0.113	0.074
Insulin	-0.073	0.331	0.088	0.436	1.000	0.197	0.185	-0.042	0.130
BMI	0.017	0.221	0.281	0.392	0.197	1.000	0.140	0.036	0.292
DPF	-0.033	0.137	0.041	0.183	0.185	0.140	1.000	0.033	0.173
Age	0.544	0.263	0.239	-0.113	-0.042	0.036	0.033	1.000	0.238
Outcome	0.221	0.466	0.065	0.074	0.130	0.292	0.173	0.238	1.000

The table 5.1, shows the data for correlation between attributes of dataset. Correlation means the degree to which to variables movements is associated. If the value is positive the relationship is positive and vice versa for neagtive. It is a no relationship when correlation is zero .

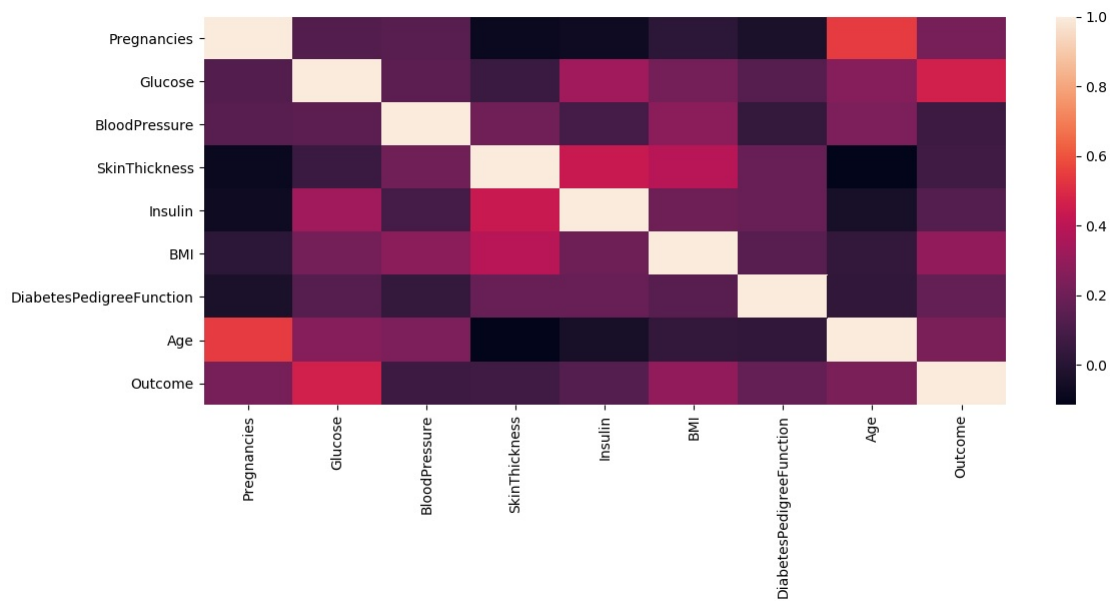


Figure 5.2: Correlation Matrix Between Attributes

The figure 5.2 shows, the correlation of data in the visualization form as described on the right side the value of color.

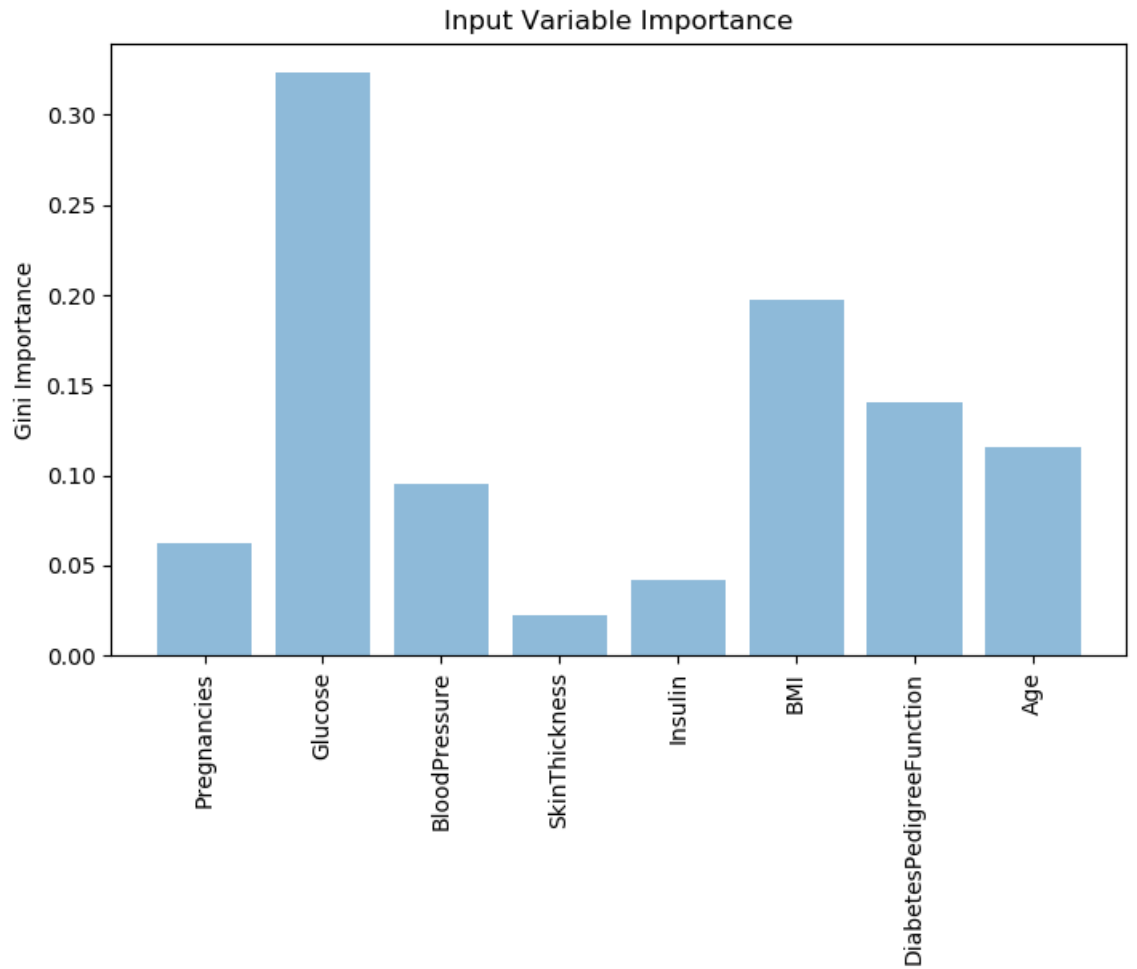


Figure 5.3: Gini index

The figure 5.3 shows the Gini Index vs. attributes graph which shows how much important attributes are. We have found that the Glucose, BMI and Diabetes Pedigree Function are the essential attributes according to the dataset.

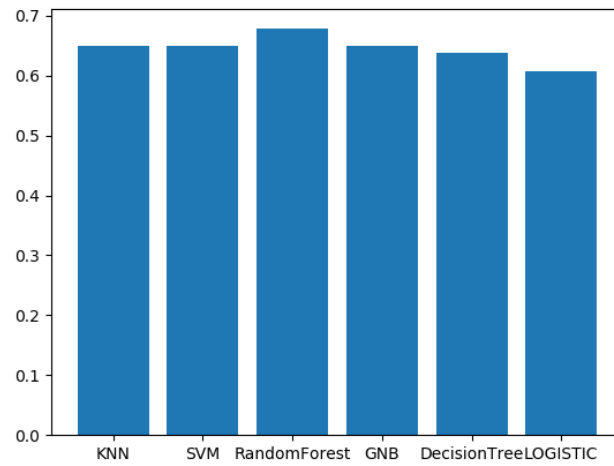


Figure 5.4: Supervised Techniques Performance Histogram

Table 5.2: Supervised Results

In the figure 5.4, the performance of various supervised techniques is shown.

Supervised Technique	Accuracy
K-NN	64.91%
SVM	65.91%
Random Forest	67.80%
Gaussian Naive Bayesian	64.91%
Decision Tree	63.86%
Logistic Regression	60.71%

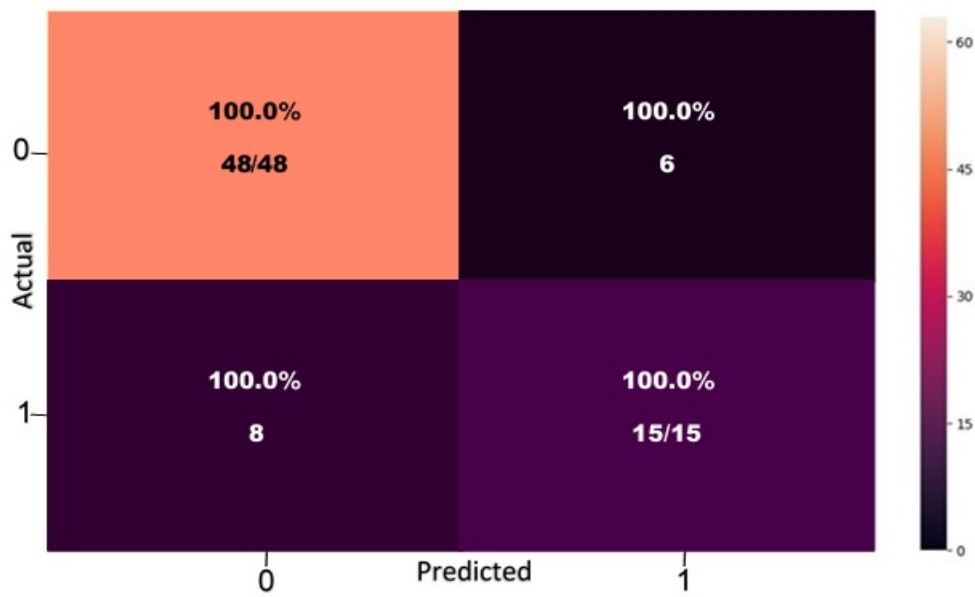


Figure 5.5: HMM using Viterbi

In figure 5.5 is the confusion matrix according to the values where true positive(tp) is 15(suffer from diabetes), and true negative(tn) is 48(No diabetes). The accuracy calculated by the sum of (tp+tn)/77(total number of patients tested) the accuracy is 81.81%.

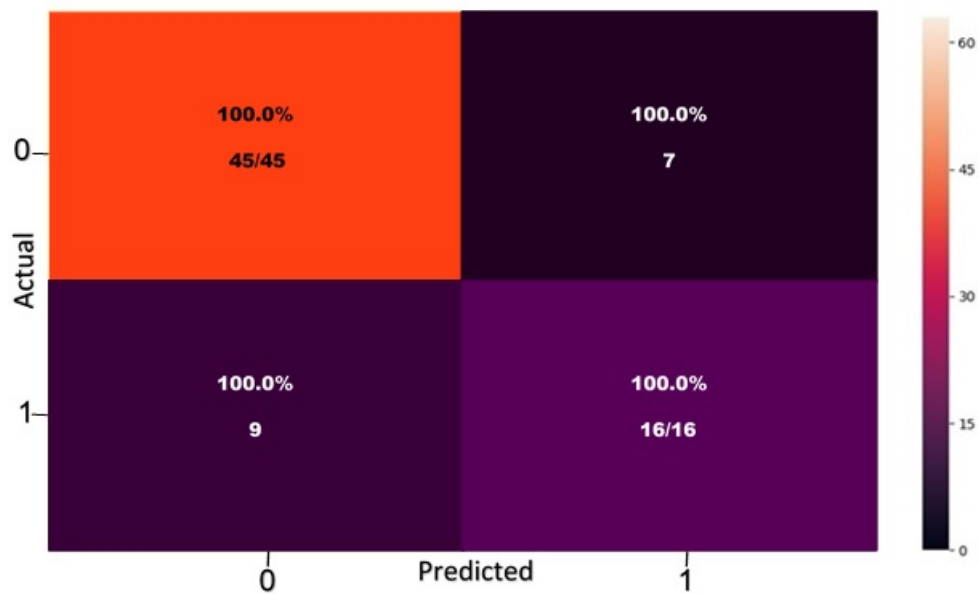


Figure 5.6: HMM using Best First

In figure 5.6 is the confusion matrix according to the values where true positive(tp) is 16(suffer from diabetes), and true negative(tn) is 45(No diabetes). The accuracy calculated by the sum of (tp+tn)/77(total number of patients tested) the accuracy is 79.22%.

Chapter 6

Conclusions

From the results, we conclude that our supervised HMM technique has performed better than the supervised techniques, i.e., Support Vector Machine, K-Nearest Neighbour, Random Forest, Decision Tree, Gaussian Naive Bayesian, Logistic Regression e.t.c., which highlights the fact that when we don't have an advantageous statistical attribute which can dominate our prediction result, these algorithms are not the smartest predictors.

We have many attributes which play a crucial role in our prediction model. According to our Gini index analysis, we found that Glucose, BMI, DiabetesPedigreeFunction & Age are essential attributes in this prediction. Well, our model might sometimes have a problem regarding the learning pattern related to gestational & type-2 diabetes, because gestational diabetes can further develop into type-2. It sometimes increases misclassification.

Gill et al. [9] work with fuzzy logic, Artificial Neural Network, and HMM on the same dataset have more accuracy than our model, but there are some facts concerning the attributes found by their analyses is Glucose, insulin, and body mass in comparison to ours are different, and the only common factor is glucose & Body mass. The work done in [9] have not given the stats on the confusion matrix of their model. Our model is on the supervised training of HMM, but in the [9] they have used fuzzy logic for decision rules, ANN for training and HMM Model as the final stage for prediction which makes it very complicated model for industrial implementation and is costly.

Chapter 7

Future Scope

We can use a hybrid model of supervised and unsupervised techniques like can use the Artificial Neural Network as a learning technique and a forward feed to the HMM to improve results. Fuzzy Logic with our model could also improve the results. We can also work with our model and can apply it to the gender-independent dataset. We can work on with a small scoring scheme for the pregnant women so that we can identify if she might have Type-2 diabetes or it is just gestational diabetes. Our model can integrate with the recurrent neural network used in [11] as a neural learning network.

Bibliography

- [1] Tresp, V., Briegel, T. and Moody, J., 1999. Neural-network models for the blood glucose metabolism of a diabetic. *IEEE Transactions on Neural networks*, 10(5), pp.1204-1213.
- [2] Kuo, H. S., Chang, H. J., Chou, P., Teng, L., & Chen, T. H. (1999). A Markov chain model to assess the efficacy of screening for non-insulin dependent diabetes mellitus (NIDDM). *International journal of epidemiology*, 28(2), pp. 233-240.
- [3] Tennvall, G. R., & Apelqvist, J. (2001). Prevention of diabetes-related foot ulcers and amputations: a cost-utility analysis based on Markov model simulations. *Diabetologia*, 44(11), pp. 2077-2087.
- [4] Honeycutt, A. A., Boyle, J. P., Broglio, K. R., Thompson, T. J., Hoerger, T. J., Geiss, L. S., & Narayan, K. V. (2003). A dynamic Markov model for forecasting diabetes prevalence in the United States through 2050. *Health care management science*, 6(3), pp. 155-164.
- [5] Cho, B. H., Yu, H., Kim, K. W., Kim, T. H., Kim, I. Y., & Kim, S. I. (2008). Application of irregular and unbalanced data to predict diabetic nephropathy using visualization and feature selection methods. *Artificial intelligence in medicine*, 42(1), pp. 37-53..
- [6] Shankaracharya, D. O., Samanta, S., & Vidyarthi, A. S. (2010). Computational intelligence in early diabetes diagnosis: a review. *The review of diabetic studies: RDS*, 7(4), pp. 252-254.
- [7] Priya, R., & Aruna, P. (2011). Review of automated diagnosis of diabetic retinopathy using the support vector machine. *International Journal of Applied Engineering Research*, 1(4), pp. 121-125.
- [8] Tapak, L., Mahjub, H., Hamidi, O., & Poorolajal, J. (2013). Real-data comparison of data mining methods in prediction of diabetes in Iran. *Healthcare informatics research*, 19(3), pp. 177-185.”
- [9] Gill, N. S., & Mittal, P. (2016). A Novel Hybrid Model for Diabetic Prediction using Hidden Markov Model, Fuzzy based Rule Approach and Neural Network. *Indian Journal of Science and Technology*, 9(35), pp. 192-199.
- [10] Bhatt, A., Dubey, S. K., & Bhatt, A. K. (2018). Analytical Study on Cardiovascular Health Issues Prediction Using Decision Model-Based Predictive Analytic Techniques. In *Soft Computing: Theories and Applications* (pp. 289-299). Springer, Singapore.
- [11] Sundaram, N. M. (2018). An Improved Elman Neural Network Classifier for classification of Medical Data for Diagnosis of Diabetes. *International Journal of Engineering Science*, pp. 16317-16321.

- [12] Hidden Markov Model algorithms, <http://jason2506.github.io/PythonHMM/>
- [13] Data Science through Python, www.datacamp.com
- [14] Bengio, Y., De Mori, R., Flammia, G., & Kompe, R. (1992). Global optimization of a neural network-hidden Markov model hybrid. *IEEE transactions on Neural Networks*, 3(2), pp. 252-259.
- [15] Marton, F. (1986). Phenomenography—a research approach to investigating different understandings of reality. *Journal of thought*, 28-49.
- [16] Seymore, K., McCallum, A., & Rosenfeld, R. (1999, July). Learning hidden Markov model structure for information extraction. In *AAAI-99 workshop on machine learning for information extraction*, pp. 37-42.
- [17] Rejeski, W. J., Ip, E. H., Bertoni, A. G., Bray, G. A., Evans, G., Gregg, E. W., & Zhang, Q. (2012). Lifestyle change and mobility in obese adults with type 2 diabetes. *New England Journal of Medicine*, 366(13), pp. 1209-1217.
- [18] Machine Learning Intro, <https://www.wikipedia.org/>
- [19] All female dataset, <https://www.kaggle.com/saurabh00007/diabetescsv/data>

Appendix A

Code for Data Analyses

This is the code for our data analyses we have used:

```
import pandas as pd
import matplotlib.pyplot as plt; plt.rcParams()
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cross_validation import train_test_split
from sklearn import tree
import seaborn as sns
import graphviz
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score

diabetes_data = pd.read_csv("diabetes.csv")
print (diabetes_data.head())

diabetes_data.quantile(q=[0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0], axis=
objects = ('0-2_preg', '3-5_preg', '>5_preg')
y_pos = np.arange(len(objects))
percentage = (40,30,30)
plt.bar(y_pos, percentage, align='center', alpha=0.5)
plt.xticks(y_pos, objects)
plt.ylabel('Percentage')
plt.title('Number_of_pregnancies')

plt.show()

print (diabetes_data.corr())

# Correlation plot
corr=diabetes_data.corr()
sns.heatmap(corr)
plt.show()
```

```

# Dimensions of data
print ("Number_of_rows , columns", diabetes_data.shape)

print (diabetes_data.dtypes)

#Null values check
print (diabetes_data.isnull().sum())

# Input and output variable seperation
Y=diabetes_data.pop( 'Outcome' )
X=diabetes_data

# Stratified sampling
X_train , X_test , y_train , y_test = train_test_split( X, Y, test_size=0.2,

# Check on dimensions
print ("length_of_train", len( X_train))
print ("length_of_test", len( X_test))

# Training the model – Decision Trees
# Create a model object
tree_model = tree.DecisionTreeClassifier()
tree_model = tree_model.fit(X, Y)

acc =cross_val_score(tree_model , X, Y, scoring='accuracy')
print (acc)

recall = cross_val_score(tree_model , X, Y, scoring='recall')
print (recall)

pres= cross_val_score(tree_model , X, Y, scoring='precision')
print (pres)

roc = cross_val_score(tree_model , X, Y, scoring='roc_auc')
print (roc)

# Prediction on training data
train_pred=tree_model.predict(X_train)
#Confusion Matrix
cm =confusion_matrix(y_train , train_pred)

print (cm)

# Prediction on test data
test_pred=tree_model.predict(X_test)
#Confusion Matrix
cm1 =confusion_matrix(y_test , test_pred)

```

```

print (cm1)

rec1 = recall_score(y_test , test_pred)
print (rec1)

pec1 = precision_score(y_test , test_pred)
print (pec1)

acc1 = accuracy_score(y_test , test_pred)
print (acc1)

[X_train.columns , tree_model.feature_importances_]

# Variable Importance

objects = ('Pregnancies' , 'Glucose' , 'BloodPressure' , 'SkinThickness' , 'I
          'BMI' , 'DiabetesPedigreeFunction' , 'Age')
y_pos = np.arange(len(objects))
percentage = (0.06223452,  0.32370587,  0.09557878,  0.02290222,
0.04227272,
              0.19695477,  0.1404274 ,  0.11592371)
plt.bar(y_pos , percentage , align='center' , alpha=0.5)
plt.xticks(y_pos , objects , rotation=90)

plt.ylabel('Gini_Importance')
plt.title('Input_Variable_Importance')

plt.show()

```

Appendix B

Code For Supervised Techniques

This is the code for our data analyses we have used for implementing different supervised techniques

```
import pandas as pd
import numpy as np
import sklearn.ensemble as ske
from sklearn import cross_validation, tree
from sklearn.externals import joblib
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix, f1_score
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn import svm
from sklearn.linear_model import LogisticRegression
import seaborn as sns

def cm_analysis(y_true, y_pred, labels, ymap=None, figsize=(10,10)):
    if ymap != None:
        y_pred = [ymap[yi] for yi in y_pred]
        y_true = [ymap[yi] for yi in y_true]
        labels = [ymap[yi] for yi in labels]
    cm = confusion_matrix(y_true, y_pred, labels=labels)
    cm_sum = np.sum(cm, axis=1, keepdims=True)
    cm_perc = cm / cm_sum * 100
    annot = np.empty_like(cm).astype(str)
    nrows, ncols = cm.shape
    for i in range(nrows):
        for j in range(ncols):
            c = cm[i, j]
            p = cm_perc[i, j]
            if i == j:
                s = cm_sum[i]
                annot[i, j] = '%.1f%%\n%d/%d' % (p, c, s)
            elif c == 0:
                annot[i, j] = ''
```

```

        else :
            annot[i, j] = '%.1f%%\n%d' % (p, c)
    cm = pd.DataFrame(cm, index=labels, columns=labels)
    cm.index.name = 'Actual'
    cm.columns.name = 'Predicted'
    fig, ax = plt.subplots(figsize=figsize)
    sns.heatmap(cm, annot=annot, fmt='', ax=ax)

data = pd.read_csv('diabetes.csv', sep=',')
print(data.head(10))
X = data.values
y = data['Outcome'].values

X_train, X_test, y_train, y_test = cross_validation.train_test_split(X, y

#Algorithm comparison
algorithms = {
    "DecisionTree": tree.DecisionTreeClassifier(),
    "RandomForest": ske.RandomForestClassifier(),
    "GNB": GaussianNB(),
    "KNN" : KNeighborsClassifier(),
    "LOGISTIC" : LogisticRegression(),
    "SVM" : svm.SVC()
}

results = {}
print("Algorithm_Test")
for algo in algorithms:
    clf = algorithms[algo]
    clf.fit(X_train, y_train)
    score = clf.score(X_test, y_test)
    print("%s : %f%%" % (algo, score*100))
    results[algo] = score
D=results
plt.figure();
plt.bar(range(len(D)), list(D.values()), align='center')
plt.xticks(range(len(D)), list(D.keys()))
bestAlgo = max(results, key=results.get)
print(' \nBest_algorithm_is_%s_with_a_%f%%_success' % (bestAlgo, results[
clf = algorithms[bestAlgo]
res = clf.predict(X_test)
mt = confusion_matrix(y_test, res)
cm_analysis(y_test, res, clf.classes_)
FS=f1_score(y_test, res, average='macro')
plt.show()
print ( 'F1_Score_is_ : %f%%' % (FS*100))
print ( 'Confusion_Matrix' )

```

```

print (mt)
print("False _positive _rate _: _%f _%%" % ((mt[0][1] / float(sum(mt[0])))*100)
print('False _negative _rate _: _%f _%%' % ( (mt[1][0] / float(sum(mt[1])))*100)

winner = max(results , key=results.get)
print(' \nWinner _algorithm _is _%s _with _a _%f _%% _success' % (winner , results[

print(' Saving _algorithm ')
joblib.dump(algorithms[winner], 'classifier.pkl')

clf = algorithms[winner]
res = clf.predict(X_test)
mt = confusion_matrix(y_test , res)
print("False _positive _rate _: _%f _%%" % ((mt[0][1] / float(sum(mt[0])))*100)
print('False _negative _rate _: _%f _%%' % ( (mt[1][0] / float(sum(mt[1])))*100)

```


Appendix C

Code For HMM Library

This is the HMM library code:

```
"""Hidden Markov models (HMMs) with supervised training."""
```

```
import numpy as np
from scipy.misc import logsumexp

from .base import BaseSequenceClassifier
from ._utils import atleast2d_or_csr, count_trans, safe_sparse_dot
```

```
class MultinomialHMM(BaseSequenceClassifier):
    """First-order hidden Markov model with multinomial event model.

    Parameters
    -----
    decode : string, optional
        Decoding algorithm, either "bestfirst" or "viterbi" (default).
        Best-first decoding is also called posterior decoding in the HMM
        literature.

    alpha : float
        Lidstone (additive) smoothing parameter.
    """

    def __init__(self, decode="viterbi", alpha=.01):
        self.alpha = alpha
        self.decode = decode

    def fit(self, X, y, lengths):
        """Fit HMM model to data.

        Parameters
        -----
        X : {array-like, sparse matrix}, shape (n_samples, n_features)
            Feature matrix of individual samples.
```

y : array-like , shape (n_samples ,)
Target labels .

lengths : array-like of integers , shape (n_sequences ,)
Lengths of the individual sequences in X, y. The sum of these
should be n_samples .

Notes

Make sure the training set (X) is one-hot encoded; if more than one
feature in X is on, the emission probabilities will be multiplied

Returns

self : MultinomialHMM
"""

```
alpha = self.alpha
if alpha <= 0:
    raise ValueError("alpha should be >0, got {}".format(alpha))

X = atleast2d_or_csr(X)
classes, y = np.unique(y, return_inverse=True)
lengths = np.asarray(lengths)
Y = y.reshape(-1, 1) == np.arange(len(classes))

end = np.cumsum(lengths)
start = end - lengths

init_prob = np.log(Y[start].sum(axis=0) + alpha)
init_prob -= logsumexp(init_prob)
final_prob = np.log(Y[start].sum(axis=0) + alpha)
final_prob -= logsumexp(final_prob)

feature_prob = np.log(safe_sparse_dot(Y.T, X) + alpha)
feature_prob -= logsumexp(feature_prob, axis=0)

trans_prob = np.log(count_trans(y, len(classes)) + alpha)
trans_prob -= logsumexp(trans_prob, axis=0)
self.coef_ = feature_prob
self.intercept_init_ = init_prob
self.intercept_final_ = final_prob
self.intercept_trans_ = trans_prob
self.classes_ = classes
return self
```

Appendix D

Main Code

This code utilizes HMM library to compute results:

```
import pandas as pd
import numpy as np
from sklearn import cross_validation
from sklearn.metrics import confusion_matrix , f1_score
import matplotlib.pyplot as plt
import seaborn as sns
from seqlearn import hmm

def cm_analysis(y_true , y_pred , labels , ymap=None , figsize=(10,10)):
    if ymap != None:
        y_pred = [ymap[yi] for yi in y_pred]
        y_true = [ymap[yi] for yi in y_true]
        labels = [ymap[yi] for yi in labels]
    cm = confusion_matrix(y_true , y_pred , labels=labels)
    cm_sum = np.sum(cm, axis=1, keepdims=True)
    cm_perc = cm / cm_sum * 100
    annot = np.empty_like(cm).astype(str)
    nrows , ncols = cm.shape
    for i in range(nrows):
        for j in range(ncols):
            c = cm[i , j]
            p = cm_perc[i , j]
            if i == j:
                s = cm_sum[i]
                annot[i , j] = '%.1f%%\n%d/%d' % (p , c , s)
            elif c == 0:
                annot[i , j] = ''
            else:
                annot[i , j] = '%.1f%%\n%d' % (p , c)
    cm = pd.DataFrame(cm, index=labels , columns=labels)
    cm.index.name = 'Actual'
    cm.columns.name = 'Predicted'
    fig , ax = plt.subplots(figsize=figsize)
    sns.heatmap(cm, annot=annot , fmt='', ax=ax)
```

```

data = pd.read_csv('diabetes1.csv', sep=',')
print(data.head(10))
X = data.values
y = data['Outcome'].values

X_train, X_test, y_train, y_test = cross_validation.train_test_split(X, y
clf = hmm.MultinomialHMM(decode='bestfirst', alpha=0.35)
#alpha=0.15
#print(alpha)
clf.fit(X_train, y_train, 8)
res = clf.predict(X_test)
mt = confusion_matrix(y_test, res)
score = clf.score(X_test, y_test)
print('Accuracy is (%):_:_')
print(score*100)
cm_analysis(y_test, res, clf.classes_)
FS=f1_score(y_test, res, average='macro')
plt.show()
print ('F1_Score is _:_:_%f_%%'% (FS*100))
print ('Confusion_Matrix')
print (mt)
print("False_positive_rate:_:_:%f_%%" % ((mt[0][1] / float(sum(mt[0])))*100)
print('False_negative_rate:_:_:%f_%%' % ( (mt[1][0] / float(sum(mt[1])))*100)

```

PLAGIARISM RESULT

Thesis

ORIGINALITY REPORT

2%

SIMILARITY INDEX

2%

INTERNET SOURCES

0%

PUBLICATIONS

%

STUDENT PAPERS

PRIMARY SOURCES

1

www.slideshare.net

Internet Source

2%

Exclude quotes Off

Exclude bibliography On

Exclude matches < 1%