Enhancement of Naïve Bayes (NB) using Packaged Hidden Naïve Bayes (PHNB)

Darpan Patel
Computer Science
Illinois Institute of Technology
Chicago, IL, USA
dpatel96@hawk.iit.com

Nisha Patel
Computer Science
Illinois Institute of Technology
Chicago, US
npate105@hawk.iit.edu

***Abstract*—Naive Bayes classifier resulting in a very good performance is based on the assumption of conditional independence between the features, which fails to give performance when there's a strong correlation between features in some datasets. The Hidden Naïve Bayes classifier gives significantly high performance, for which base line is to store the correlation in the hidden parent between all the pairs of features. The reason behind enhancing to Packaged Hidden Naïve Bayes (PHNB) classifier is that it takes too long testing time for higher dimensions of features. Central idea of the PHNB algorithm is to control over hidden packets through packaging. We have experimented this and the result shows**

***Keywords*—Naïve Bayes; Hidden Naïve Bayes (HNB); Packaged Hidden Naïve Bayes (PHNB); Performance; test time; classification.**

I. Introduction

Naïve Bayes classifier, is a standard classifier used for classification, which is constructed by training the train dataset having class label with it. For a data matrix X assume having A_1, A_2, A_3, …, A_n features of which a vector E $<a_1,a_2,a_3,\dots,a_n>$ is an instance, with *m* instances in all. C is class label having different class labels *c* for the given dataset. So c(E) is the predicted label for any given vector of n features belonging to C. Bayesian classifier is defined as:

$$c(E) = argmax_{c \in C} P(c)P(a_1,a_2,a_3,\dots,a_n|C) \quad (1)$$

Now, assuming that all the vectors are independent of each other and with the give true class labels,

$$P(E|c) = P(a_1,a_2,a_3,\dots,a_n|C) = \Pi_{i=1 \text{ to } n} P(a_i|C) \quad (2)$$

So, the resulting Naïve Bayes (NB) classifier can be defined as:

$$c(E)=argmax_{c \in C} P(c) \Pi_{i=1 \text{ to } n} P(a_i|C) \quad (3)$$

NB classifier is very easy to implement, simple and efficient until the assumptions are satisfied. Its performance drops significantly and becomes very poor when the assumption not being satisfied in the real world datasets always. Many datasets have features very much correlated to each other and because of the baseline of the algorithm assumption of conditional independence which gets violated in this scenario and results into bad accuracy. Fig. 1 shows graphically the structure of naive Bayes [2]

Naive Bayes is the simplest form of Bayesian networks. It is obvious that the conditional independence assumption in naive Bayes is rarely true in reality, which would harm its performance in the applications with complex attribute dependencies.

After facing this problem with Naïve Bayes algorithm, in past 2 decades a number of algorithms have been designed and various classifiers have been built. This comprises of Structural extension, which explains the dependencies of attributes and because of which the learning of Bayesian structure is inevitable. Finding optimal solution for it is a NP-Hard Problem. Many other like TAN, ODANA, AODE and WAODE having some or the other differences from each other has been studied and implemented.

Extension of structure of Naïve Bayes is the easiest way to overcome from the limitation of naïve Bayes. Tree - Augmented naïve Bayes(TAN) is an extended tree-like naive Bayes in which the class node directly points to all attribute nodes and an attribute node can have only one parent from another attribute node. Fig 2 shows the structure of TAN algorithm. TAN is stepwise evolution of NB and another structural form of Baysian Network.

In TAN classifier, class node C directly points to all attributes nodes but no limitation for arcs among the attributes nodes except they do not make any directed cycle. In learning of TAN consider only one attribute parent for each attribute and ignore the influences from other attributes.

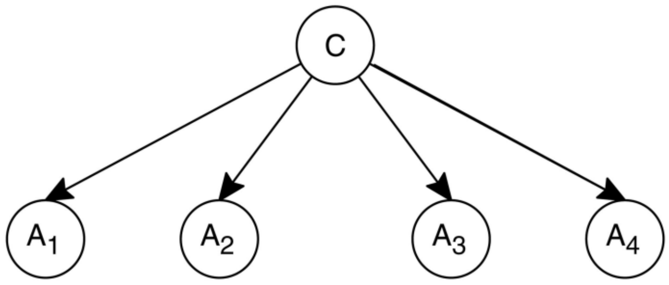


Fig. 1. An example of naive Bayes (NB) [2]

The performance of the classifier is measured by its classification accuracy. Here, We represent the other classification parameter like Precision, Recall, F-score etc. In above algorithm the only classification is in consideration for accuracy, but there are some other applications in which misclassification cost should be taken in consideration.

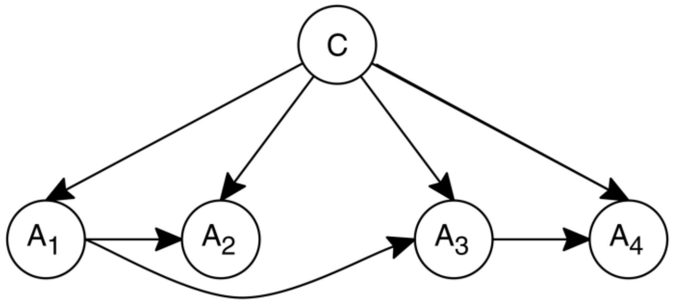


Fig. 2. An example of TAN [2]

In this report, we first summarize the performance of Naïve Bayes Binomial on different datasets of UCI[3] . Accuracy is falling for mutually dependent datasets so that proposed the algorithm Hidden Naïve Bayes(HNB). In HNB, a hidden parent is created for each attribute which combines the influences from all other attributes. Followed by the description of our experimental setup and results in detail, we make a conclusion about HNB. When HNB applied on high dimensional dataset it takes too much training time as well testing time because all attributes are concerned with the hidden parent.

There may be possibility that all attributes are not correlative to all attributes so, the number of correlative attributes with a specific attribute is further lower than the number of all attributes, especially when the dataset dimension is high. Now, Package Hidden Naïve Bayes comes in the picture and that controls the number of correlative attributes to find the optimal solution of HNB model packaged with NB model. Finally, We conclude with the optimal solution of PHNB on various datasets of UCI which has type of numerical values for k-class classification.

Measure the Accuracy and total time (training + testing) and compare both HNB and PHNB. PHNB significantly reduces the testing time but it's training time is same as HNB. It's overcome the drawback of HNB and increase the accuracy of it. Finally, we conclude this report and propose the future work.

II. Problem Statement

Naïve Bayes algorithm worked very well on datasets when there is no internal dependencies of features on each other. But the problem arises when the features are correlated and have dependencies. In such case, Naïve Bayes classifier fails to give the accuracy and shows a very bad performance.

As a result, a number of solutions were proposed and many classifiers comes into picture. So our main focus here was to attempt to implement Hidden Naïve Bayes and Packaged Hidden Naïve Bayes classifier an extension of Hidden Naïve Bayes. Hidden Naïve Bayes is designed such that it correlates the features and stores them in hidden parents. We want to implement this classifier and evaluate its performance in terms of accuracy, classification and testing time.

Packaged Hidden Naïve Bayes algorithm shows a very good behavior in terms of accuracy and testing time as compared to those of Hidden Naïve Bayes and Naïve Bayes. So to measure both accuracy and time measurements, and hence its algorithm is to be implemented.

For implementing both of these classifiers, there was a need to implement Naïve Bayes classifier as well, which is in turn used from sklearn as the central idea here was to implement Hidden naïve Bayes and Packaged Hidden Naïve Bayes classifiers.

III. Proposed Solution

For implementing both of this classifier it was necessary to implement the inbuilt Naïve Bayes Classifier. In the solution we have followed the sequence and algorithm provided in the reference paper.

A. Hidden Naïve Bayes Classifier:

Of all the latest research topics HNB is the youngest in the family of Naïve Bayes in which the dependencies between all the pairs of attributes is hidden in the parent of each attribute. The parent containing all the hidden attribute's dependencies a_{hpi} , and hence can be defined as:

$$c(E)=\arg\max_{c\in C} P(c)\prod_{i=1 \text{ to } n} P(a_i | a_{hpi}, C) \quad (4)$$

For the above equation, the results can be calculated by calculating parameters Wij which defines the weight between attributes Ai and Aj.

$$W_{ij} = \frac{Ip(A_i;A_j|C)}{\sum_{j=1,j\neq i}^{n} Ip(A_i;A_j|C)} \quad (5)$$

The conditional mutual information between attributes is defined as:

$$Ip(A_i;A_j|C) = \sum_{a_i,a_j,c} P(a_i,a_j,c)\log\frac{P(a_i,a_j|c)}{P(a_i|c)P(a_j|c)} \quad (6)$$

The conditional probability of the hidden parent and attributes is shown as:

$$P(a_i|a_{hpi},c) = \sum_{j=1,j\neq i}^{n} W_{ij}\times P(a_i|a_j,c) \quad (7)$$

In this algorithm all the attributes are paired for each example and then their condition probability information is calculated, which gives us the probability of how frequently does a pair of attributes occur over a given a pair of attributes over all the examples given. Then weights of the attributes are calculated which helps to calculate the average probability of conditional probability information foe a given pair of attributes. Then the conditional probability is calculated individually for each attribute. The prediction is then made for any given example by calculating the posterior class probability and its corresponding conditional probability over all the attributes. This sequence is followed for all the different class and then and then max value is chosen and corresponding classes is selected and predicted for that given vector.

Algorithm (HNB) :

```
Input: a set D of training examples
Output: a hidden naive Bayes for D
    for each value c of C
        Compute $P(c)$ from D
    for each pair of attributes Ai and Aj
```

```
        for each assignment ai; aj, and c to Ai;Aj, and C
                Compute $P(ai|aj,c)$ from D

    for each pair of attributes Ai and Aj
        Compute $Ip(Ai ; Aj | C)$ and $W_{ij}$ from D
```

Implementation Details:

1) Write a separate program in python named HNB.py contain several functions of training phase as well as testing phase.

Training phase :

a) Calculate prior probability of each class, conditional probability which are used in calculate of Conditional Mutual Information.

b) Calculate conditional probability which are used in calculation of hidden parent probability.

c) I used mainly dictionary structure to store all prior probability so that I can easily access at the time of testing phase.

d) Calculate conditional mutual information for all pair and each assignment and store it as a Matrix of IP.

e) Calculate the Weight matrix Wij based on the IP matrix.

2) Testing phase do classification based on probability of each attribute to its hidden parent probability and given class.

a) For each example find the probability of attribute to its hidden parent for each class.

b) Based on maximum membership of any class make prediction of particular example

3) Find accuracy and Total time to run whole program. Calculate all parameter named Precision, Recall and F-score.

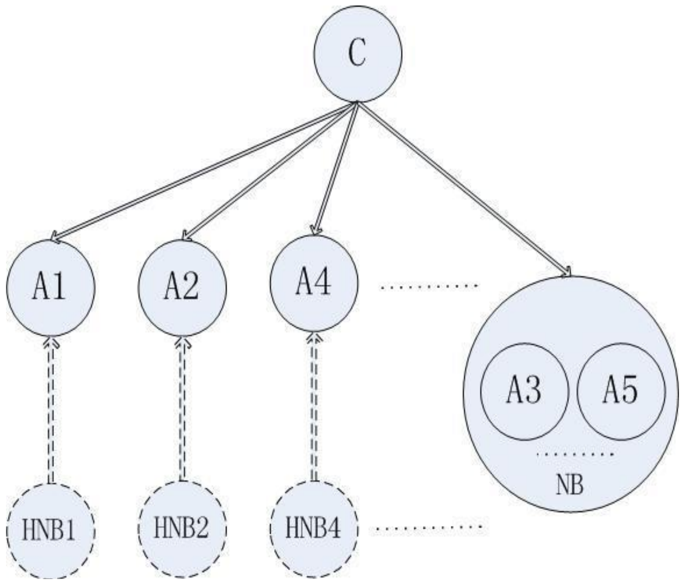
B. Packaged Hidden Naïve Bayes

Packaged Hidden naïve Bayes Algorithm, an extension of Hidden Naïve Bayes and Naïve Bayes is considered as a better approach when there are a large number of feature for a given dataset.

The central idea of this classifier is to calculate conditional probability information for all the given pair of attribute and while classification there is a change in the flow of algorithm, a new parameter threshold is calculated. This parameter is compared to conditional information for a given pair of attribute and is checked and if the value is greater than threshold then the the attributes are placed in the bag of that given attribute or else then a bag of Naïve Bayes is created and the attribute are placed in that bag.

Now during testing, for a given example its conditional probability is calculated using two different approaches, Naïve Bayes and Hidden Naïve Bayes. The attributes of example are compared to those with bags. The attributes having individual bag are calculated using the Hidden Naïve Bayes and the ones falling in individual bags are calculated using Naïve Bayes.

The following image explains this idea:



The classifier for PHNB can be defined by the following equation:

$$c(E) = \arg\max_{c \in C} P(c) \prod_{i=1}^{number of\ HNB bags} P(a_i \mid a_{hpi}, c) \prod_{j=1}^{number of\ attributes in\ NB} P(a_j \mid c) \quad (8)$$

The conditional probability of a given attribute and its parent is defined in the following equation:

$$P(a_i \mid a_{phi}, c) = \sum_{j=1, j\neq i}^{number of\ attributes in\ HNBi} W_{ij} \times P(a_i \mid a_j, c) \quad (9)$$

Weights of pair of attributes is calculated using the following equation:

$$W_{ij} = \frac{Ip(A_i; A_j \mid C)}{\sum_{i=1, j\neq i}^{number of\ attributes in\ HNNBi} Ip(A_i; A_j \mid C)} \quad (10)$$

The new value, Threshold is calculated using the following equation:

$$average = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} Ip(A_i; A_j \mid C) \quad (11)$$

Threshold can be calculated with two different approaches mentioned below:

- computing the average of dependences of all pairs of attributes
- empirical value

Threshold can be chosen depends upon the dataset. If dataset has high dimensional features, then it must be bigger value so that more attribute can calculate using NB rather than HNB bag.
Algorithm (PHNB):

```
Training phase: (Input: a set $D$ of training examples)
    For each value $c$ of $C$
        Compute $P(c)$ from $D$;
    For each pair of $Ai$ and $Aj$ in $D$
        For each assignment $ai$, $aj$ and $c$ to $Ai$,$Aj$ and $C$
        Compute $P(ai|aj,c)$ from $D$;
    For each pair of attributes $Ai$ and $Aj$
        Compute $Ip(Ai ; Aj | C)$ from $D$;
    For each $Ai$
        For each $Aj \neq Aj$ in D
        If $Ip(Ai;Aj|C) \geq$threshold, put $Aj$ into the bag HNBi;
        else put $Ai$ into the bag NB;
Fo
Testing phase: (Input: an instance $E$)
    For each value $c$ of $C$
        For each $Ai$ in E
        If HNBi exists, computed by the equation 9;
        Apply NB algorithm in the bag NB
        Compute $c(E)$;
```

Implementation Details:

1) Write a separate program in python named HNB.py contain several functions of training phase as well as testing phase.

Training phase :

a) Calculate prior probability of each class, conditional probability which are used in calculate of Conditional Mutual Information. Initial probability phase is same as HNB.

b) After calculating IP need to check correlative attribute and independent attributes. For that introduce Threshold value based on IP value mentioned in equation.

c) Threshold can be find using that equation(11) or we can take another approach of take optimal value according to dataset.

d) According to IP and threshold value separate all attributes into HNB bag and NB bag. We made two functions and dictionary structures to Make HNB bags and NB bags using IP matrix. Use this Bags at the time of testing phase.

Testing Phase:

For each example calculate the hidden parent probability for attributes which are in HNB bags and return the product value for each attributes.

Find the conditional probability of the attributes which are in NB bags and return the product of each attributes.

Find membership of example for all classes using max value of each classes then predict the "y" value using user-defined functions.

Parameter Calculation:

Find Accuracy, Precision , Recall and F-score and compare to other datasets.

IV. Result and Discussion

We used different datasets of UCI - machine learning. Here, we considered dataset which has numerical values for multi-class classification.

Dataset :

1) IRIS Dataset
2) Balance-scale Dataset
3) Breast-cancer-wisconsin dataset
4) Heart-statog Dataset
5) Mfeat-zer (Difficult to implement)

Description of dataset is mentioned on UCI website and we provide link in reference part as well. Datasets links is already used in program as well as
mentioned in reference at end of report.

All Dataset contain different no of examples and features so accuracy and total time is different for every datasets.

We used same dataset for In-Built Naïve Bayes algorithm, Hidden Naïve Bayes (HNB) and Package Hidden Naïve Bayes (PHNB).

Accuracy and Other parameter after mentioned below in the table. All results obtained are for 10-Fold validations.

IRIS (NB)	Accuracy	Precision	Recall	F-score
0 – class	1.0	1.0	1.0	1.0
1 - class		1.0	1.0	1.0
2 - class		1.0	1.0	1.0

IRIS(HNB)	Accuracy	Precision	Recall	F-score
0 – class	0.933	1.0	0.8889	0.9412
1 – class		0.8571	1.0	0.9231
2 – class		0	0	0

IRIS(PHNB)	Accuracy	Precision	Recall	F-score
0 – class	0.8667	1.0	1.0	1.0
1 - class		1.0	0.333	0.5
2 - class		0.8182	1.0	0.9

HEART(NB)	Accuracy	Precision	Recall	F-score
0 – class	0.8519	0.7333	1.0	0.8462
1 - class		0.7219	0.9423	0.8672

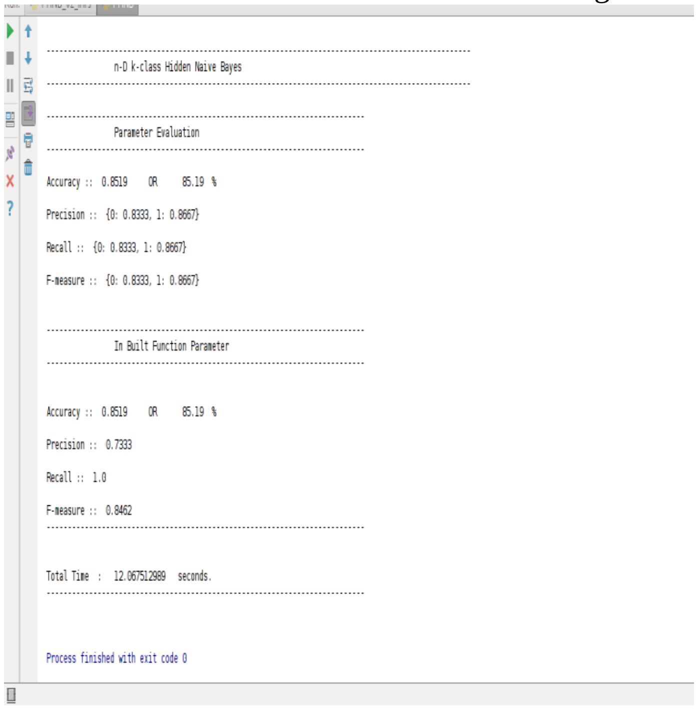
HEART(HNB)	Accuracy	Precision	Recall	F-score
0 – class	0.8519	0.8333	0.8333	0.8333
1 - class		0.8667	0.8667	0.8667

HEART(PHNB)	Accuracy	Precision	Recall	F-score
0 – class	0.8312	0.8426	0.8426	0.8426
1 - class		0.8792	0.8792	0.8792

Time taken for computation of PHNB

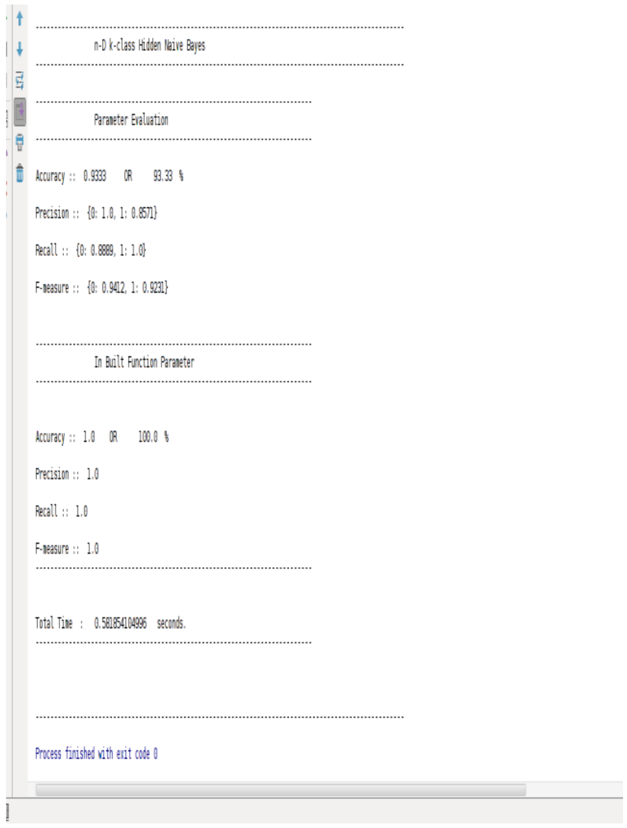
	Total Time(sec)	Testing Time(sec)
IRIS Dataset	0.532	0.034
Balance-scale	11.239	0.39
Breast-cancer-wisconsin	98.4	1.13
Heart-statog	12.98	0.5
Mfeat-zer	Very high	2.4(2-Fold)

Screenshot for result obtained for Heart- statlog:



```
--------------------------------------------------------------------------------
          n-D k-class Hidden Naive Bayes
--------------------------------------------------------------------------------

------------------------------------------------------------
          Parameter Evaluation
------------------------------------------------------------

Accuracy ::  0.8519    OR     85.19 %

Precision ::  {0: 0.8333, 1: 0.8667}

Recall ::  {0: 0.8333, 1: 0.8667}

F-measure ::  {0: 0.8333, 1: 0.8667}


------------------------------------------------------------
          In Built Function Parameter
------------------------------------------------------------


Accuracy ::  0.8519    OR     85.19 %

Precision ::  0.7333

Recall ::  1.0

F-measure ::  0.8462
------------------------------------------------------------


Total Time  :   12.067512989   seconds.
------------------------------------------------------------



Process finished with exit code 0
```

Screenshot for result obtained for IRISH Dataset:



```
--------------------------------------------------------------------------------
            n-D k-class Hidden Naive Bayes
--------------------------------------------------------------------------------

------------------------------------------------------------
            Parameter Evaluation
------------------------------------------------------------

Accuracy ::  0.9333    OR     93.33  %

Precision ::  {0: 1.0, 1: 0.8571}

Recall ::  {0: 0.8889, 1: 1.0}

F-measure ::  {0: 0.9412, 1: 0.9231}


------------------------------------------------------------
            In Built Function Parameter
------------------------------------------------------------


Accuracy :: 1.0    OR     100.0 %

Precision ::  1.0

Recall :: 1.0

F-measure ::  1.0
------------------------------------------------------------


Total Time  :   0.581854104996   seconds.
------------------------------------------------------------




--------------------------------------------------------------------------------

Process finished with exit code 0
```

Discussion:

- Here, We mentioned the total time which consist the training time and testing time.
- Here training time is little bit high for higher dimensional datasets like "Mfeat-zer" because of complexity of algorithm O($kn^2v^2 + tn^2$).
- Testing time is accurate for PHNB but its little bit high for HNB.
- Average accuracy for HNB is 85-90 % for medium datasets which have features between (20-50). For higher dimensional datasets accuracy is around 70%-80%. Deviation of HNB accuracy is 2.5-3.5
- Average accuracy for PHNB is around 92-93 % for medium and higher dimensional datasets. Deviation is flicker around 1.5-2.5.
- So, PHNB is better in terms of accuracy as well as test time.

We have faced a lot of difficulties for running the implementations of HNB and PHNB. The run time complexity of

Conclusion and Future Work

The PHNB model results in to higher performance while testing phase using both the HNB and NB models. The PHNB model is both a combination of NB and HNB as a result of which again it takes a lot of time in training any higher dimension dataset.

For HNB, the computation complexity is very high. There is a lot of overhead in calculating the conditional probability information as it is compared for each pair of attribute of each example.

There are a number of things that can be focused on for future works:

- Because of the lack higher configuration of hardware a lot of datasets with higher attributes and higher examples could not be performed over local machines. So we can optimize the algorithm such that it can work on local machines with regular configuration.
- The computation of bags can be done in parallel, so if the bags are computed parallel then the computation time of PHNB can be reduced significantly resulting into a better performance.

Acknowledgment

The research paper we used as a reference seemed to be a very good source of knowledge. It provided a lot of information regarding these (HNB and PHNB) topics and algorithm as well.

Also, would want to acknowledge Gady Agam our professor for providing very deep knowledge in this subject made us capable to attempt research topics as well.

References

During the study of this project's coursework, we have come across a number of research papers and also have gone through various websites enlightening about the topic of this paper.

For using the various datasets, UCI dataset repository have been very useful.

[1] A novel Naive Bayes model: Packaged Hidden Naive Bayes.

[2] L. Jiang, H. Zhang, and Z. Cai. A Novel Bayes Model: Hidden Naïve Bayes. In: IEEE Transactions on Knowledge and Data Engineering, Vol.21(10), pp. 1361-1371, 2009.

[3] Increasing the Accuracy of Hidden Naive Bayes Model Sotiris Kotsiantis 1 ,Vasilis Tampakas 2 TEI of Patras, 1 M. Alexandrou str., Koukouli, 263 34 Patras, Greece

[4] L. Jiang, D. Wang, Z. Cai and X. Yan : Survey of Improving Naive Bayes for Classification. In: Lecture Notes in Computer Science, Vol.4632, pp. 134-145, Springer-Verlag , Berlin Heidelberg , 2007 .

[5] http://sourceforge.net/projects/weka/files/datasets/UCI%20and%20StatLib/uci-0070111.tar.gz//download

[6] https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data

[7] https://archive.ics.uci.edu/ml/machine-learning-databases/balance-scale/balance-scale.data

[8] https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.data

[9] https://archive.ics.uci.edu/ml/machine-learning-databases/statlog/heart/heart.dat

[10] https://archive.ics.uci.edu/ml/machine-learning-databases/mfeat/mfeat-zer

[11] http://archive.ics.uci.edu/ml/