Impact of Different Data Types on Classifier Performance of Random Forest, Naïve Bayes, and K-Nearest Neighbors Algorithms

Asmita Singh, Malka N. Halgamuge, Rajasekaran Lakshmiganthan
School of Computing and Mathematics
Charles Sturt University
Melbourne, Australia

***Abstract*—This study aims to evaluate impact of three different data types (Text only, Numeric Only and Text + Numeric) on classifier performance (Random Forest, *k*-Nearest Neighbor (*k*NN) and Naïve Bayes (*NB*) algorithms). The classification problems in this study are explored in terms of mean accuracy and the effects of varying algorithm parameters over different types of datasets. This content analysis has been examined through eight different datasets taken from UCI to train models for all three algorithms. The results obtained from this study clearly show that RF and *k*NN outperform NB. Furthermore, *k*NN and RF perform relatively the same in terms of mean accuracy nonetheless *k*NN takes less time to train a model. The changing numbers of attributes in datasets have no effect on Random Forest, whereas Naïve Bayes mean accuracy fluctuates up and down that leads to a lower mean accuracy, whereas, *k*NN mean accuracy increases and ends with higher accuracy. Additionally, changing number of trees has no significant effects on mean accuracy of the Random forest, however, the time to train the model has increased greatly. Random Forest and *k*-Nearest Neighbor are proved to be the best classifiers for any type of dataset. Thus, Naïve Bayes can outperform other two algorithms if the feature variables are in a problem space and are independent. Besides Random forests, it takes highest computational time and Naïve Bayes takes lowest. The *k*-Nearest Neighbor requires finding an optimal number of *k* for improved performance at the cost of computation time. Similarly, changing the number of attributes that effect Naïve Bayes and *k*-Nearest Neighbor performance nevertheless not the Random forest. This study can be extended by researchers who use the parametric method to analyze results.**

***Keywords*—Big data; random forest; Naïve Bayes; k-nearest neighbors algorithm**

I. Introduction

The public health sector, science laboratory, retail, and banking, heavily rely on the internet for their daily interactions with customers: the amount of data obtained is huge and growing rapidly ranging from Terabytes to Petabytes, known as Big Data. Big data is practically reshaping all business sectors [1], as it is a great source of advancement and economic value, thus analyzing Big Data leads to better insights and new understanding in assisting of different sectors for better decision making. Machine learning is a technique used to be known as big data that assists to get deep insights [2], defined as the process of discovering the relationships between predictor and response variables using computer-based statistical approaches [3]. There are different kinds of statistical approach or methods are used in machine learning. However, the Supervised learning approach is one of the prominent approaches which trains sets of data with labeled classes to train the models known as a classifier based on features or attributes [4]. This model can be used to predict class label (discrete value) of any new data instance. There are several learning algorithms that use this approach to classify objects or data instances into two or more labels such as Support Vector Machines, Logistic Regression, Decision Trees, Random Forest, Naïve Bayes, etc.

A classifier performance largely depends on characteristics of classified data sets. Various comparisons have been made on different classifier performance over various datasets to find suitable classifier for a given problem. Even with high performing computers solving complex problems requires most suitable classification techniques to avoid wastage of time and resources. Prediction in health sector requires greater degree of precision for improved diagnosis and treatment, whereas areas such as disaster management requires less computation time in prediction to take actions timely saving lives. This paper discusses and provides a comparative study of supervised classification algorithms on different types of Big Data sets. These sets comprise of Random Forest, *k*-Nearest Neighbors, and Naïve Bayes classification algorithms are tested and measured not only their efficiency and processing time performance nonetheless also observes algorithm behavior on the various datasets feature spaces. The accuracy of the estimation obtained gives a clear picture of algorithm performance over different data type and feature dimensions. This study uses standard k-fold cross-validation to obtain reliable estimates [5].

This paper is structured as follows: Section 2 demonstrates a conceptual frame of research and uses techniques with the detailed discussion of algorithms used. Datasets selected are from different areas of application such as Medical, Banking, Commerce, Census etc. Each algorithm requires some parameter optimization as discussed in Section 2 and their effects on computation time and performances are demonstrated through experiments. The result section, outlined in Section 3 shows plots and histograms obtained as result of data prediction and parameter optimization. Section 4 discusses the experiment results in detail with any issues

faced. One of the major result obtained in this research is that Random Forest performs consistently well in all types of datasets nonetheless it takes the largest computation time. Lastly, Section 5 concludes this study.

II. Material and Methods

Classification algorithms have been used for training datasets whose classes are known to learn and create prediction. The models store the trained datasets in memory to predict while classification is called lazy-learning. This model can predict the categorical class label or group of new data instance [6]. This study has used supervised classification algorithm known as Random Forest, Naïve Bayes, and lazy-learning algorithm *k*-Nearest Neighbor to predict class labels to test data sets. The analysis is performed by using Python 3.6.0 on Mac OS X EI Capitan (Version 10.11.6), 2.4 GHz Intel Core i5, 8GB RAM, 1600 MHz DDR3 (see Appendix).

A. Mathematical Formuation

If we take the dataset S of n observations or instances with p attributes and $(p+1)^{th}$ attribute as the response or target variable y depends on p attributes. We can then combine attributes to form p-dimensional vector as

$$x = (x_1, x2, x3,. x_p). \quad (1)$$

Further, the response variable $y \in (y_1, y2, ..., y_m)$, where m is the number of distinct classes or labels. Then, the p attributes are called predictor variables and response variable y that depends on p attributes of x.

We will study the classification problems where y is categorical variable and there is scalar function f, which assigns a class or label to every such vector x as

$$y = f(x)+\in . \quad (2)$$

We call f the prediction function as we take M such vectors given together with attributes and corresponding classes as the training set.

$$x^{(i)}, y^{(i)} \text{ for } i= 1,2, ..., M. \quad (3)$$

For any new sample where $x = z$, finds the class of this sample, we assume f is sufficiently smooth and trains the chosen algorithm to learn from or to use training data sets in predicting the class or label for new sample or instance. This study will use abbreviations RF for Random Forest, *k*NN for *k*-Nearest Neighbor (*k*-NN) and NB for Naïve Bayes.

B. Algorithms

1) K-Nearest Neighbors Algorithm: The *k*-Nearest Neighbors algorithm is known as *lazy-learning* algorithm as it takes less time for training. Its computations are eager-based learning algorithms (such as Decision trees, Random forest, Naïve Bayes, etc.) and takes less time during classification [7]. The *k*NN constructs predictions directly from training dataset which is stored in the memory. To classify an unknown data, for instance, *k*NN finds the set of *k* objects from the training data closest to input data instance by a distance calculation and assigns maximum voted classes out of these neighboring classes [8].

Choosing the optimal value of k is important and effects the classifier performance. This study tries different odd values of k up to 25 on the problem space to the *k*-Nearest classifier with cross-validation and chooses the one with least misclassification error. The closeness of the new data instance with the training data instances can be measured by a number of ways, including Euclidean distance and distance Manhattan. The most widely used technique is the Euclidean distance [9]. This study uses Euclidean distance measure for *k*-Nearest Neighbors classification algorithm. Euclidean distance d between two data points x and y are calculated using the formula below [10]:

$$d(x,y) = \sqrt{(\sum_{i=1}^{N}(x_i - y_i)^2)}. \quad (4)$$

2) Random Forest Algorithm: Random Forest algorithm is an ensemble classifier algorithm which uses 'bagging' to create multiple decision trees and classifies new incoming data instance to a class or group [11]. The trees are built not pruned [12]. The name randomly comes from the selection of random n features or attributes to find the best split point using Gini-index cost function while building decision trees [10], [13]. This random selection of the predictor variables results in less correlation among the trees and has a lower error rate [14]. To predict target value for new data instance, the new observation is fed to all classification trees in the Random Forest. The numbers of prediction for a class performed by each of the classification trees are counted. Then, the class with the maximum number of votes is returned as the class label for new data instance [15].

For prediction, accuracy is important such as medical fields, processing times can be used as a trade-off, whereas the time-sensitive fields seek quick predictions such as disaster prediction percentage accuracy that can also be used as a trade-off with time. This study will show accuracy percentage and processing time differences between increasing numbers of trees per node.

3) Naïve Bayes Algorithm: Naïve Bayes algorithm uses probabilities of each attribute that belonging to each class in the training set to predict the class of new data instances. Naïve Bayes predicts datasets with the assumption that attributes belonging to a class that is independent of each other. This study uses Gaussian Naïve Bayes algorithm which works well with both continuous and discrete datasets.

Given a data instance is X, described by its feature vector $(x_1, ..., x_n)$, and a class target y, Bayes' theorem states the conditional probability $P(y|X)$ as a product of simpler probabilities using the naïve independence assumption:

$$P(y|X) = \frac{P(y)P(X|y)}{P(X)}$$

$$= \frac{P(y)\prod_{i=1}^{n}P(x_i|y)}{P(X)}. \quad (5)$$

$P(X)$ is constant, hence we classify the given data instance by finding:

$$\hat{y} = \arg max_y P(y)\prod_{i=1}^{n}P(x_i|y). \quad (6)$$

P(y) is the frequency of samples in training set with class *y* and P (x_i |y) that is calculated assuming the likelihood of features to be Gaussian [18].

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\left(-\frac{(x_i-\mu_y)^2}{2\pi\sigma_y^2}\right). \quad (7)$$

If the classifier encounters a word that has not been seen in the training set, the probability of all the classes would become zero and there would not be anything to compare with. This problem can be solved by ignoring such values.

C. Datasets

This study has used supervised learning algorithms *k*-Nearest Neighbors, Naïve Bayes and Random Forest to learn and predict class or target values of any new unseen data instances. The behaviors of these algorithms are studied for numerical datasets only, text only dataset and mix dataset as seen in Table I. The datasets chosen are a mix of big data and small size data with a varying number of attributes. Datasets used in this study are taken from UCI Machine Learning Repository [16] (Fig. 1).

D. Data Split and Validation

1) Cross-Validation: To split datasets, for training and testing purposes, the datasets will need to be evaluated. This study has used K-fold cross-validation technique to do this particular evaluation. The K-fold cross-validation splitting is a standard technique that splits the dataset into k equal parts or folds where *k-1* parts are used for training and the remaining ones for testing. This process is repeated *k* times and each time with different subsets of k. We can average the evaluated error rates of each *k* fold iteration to find average error rate [9], [15]. Through a number of studies, it has been found that the repeated cross-validation iteration over dataset converges to correct performance of respective classifier and 10-fold cross-validation is better than k-folds and leaves one out of this validation [17]. Using this method does not require to separate the list of testing or training the data that avoids problems of overfitting [10].

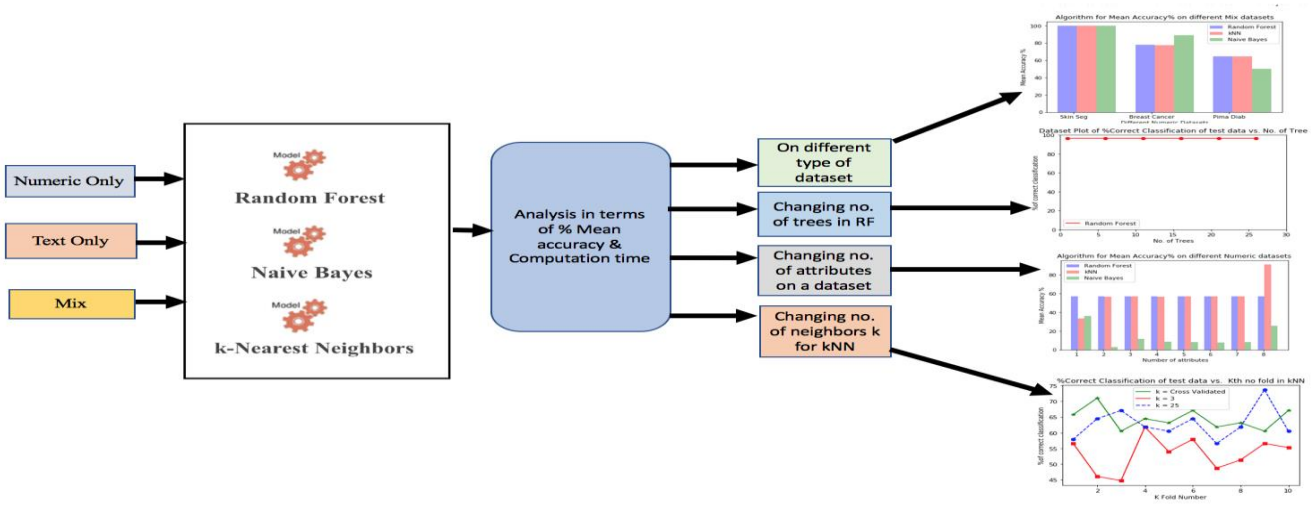

Fig. 1. Graphical abstract: Comparative study of random forest, Naïve Bayes *K*-Nearest neighbor on different types of datasets.

TABLE I. DATASET INFORMATION

Dataset No.	Dataset Type	Dataset Name	No. of Dataset Instances	No. of Dataset Attributes	Dataset Size
1	Numeric + Text	Online Retail Data Set	541909	8	45.9 MB
2	Numeric + Text	Diabetes 130-US hospitals for years 1999-2008 Data Set	100000	52	19.2 MB
3	Numeric + Text	Adult Data Set	48842	14	3.8 MB
4	Numeric Only	Skin Segmentation Data Set	245057	4	2.6 MB
5	Numeric Only	Pima Indians Diabetes Data Set	768	8	24 KB
6	Numeric Only	Breast Cancer Data Set	699	10	25 KB
7	Text Only	Nursery Data Set	12960	8	1.1 MB
8	Text Only	News Aggregator Data Set (NewsCorp)	417555	7	84.4 MB

E. Parameter Optimization

1) k for k-Nearest Neighbor: The *k*-Nearest Neighbors requires *k* value to vote the *k* number of neighbors around new test data instance and classifies it to the maximum voted class label. It has been suggested by studies that even values of *k* are not suitable as they result in draw [18] and the odd values of *k* result in higher classification accuracy than odd values [19].

To further investigate the effect on accuracy and select "optimal k" this study performs cross-validation classification on the dataset with *k*NN algorithm. In this research, the odd values $k = 1, 3, 5, 7, 9$ to 50 are used to cross validate and classify a small subset of data set. The *k* value resulting in the least misclassification error is selected to predict.

2) Number of trees for Random Forest: Random forest algorithm takes number of trees as parameter to build that many trees used to predict class by taking average or maximum vote over all trees. In this study dataset prediction is performed on different number of trees with $n\_trees = 1, 5, 10, 15, 25$. Cross-validation over different *n_trees* can give optimal number of trees nevertheless with large computation time and space in RAM.

III. Results

Various datasets are used to train and test k-fold using cross-validation and predict class labels. Missing values in the dataset are ignored and few datasets have attributes with very few values that are removed.

Finally, complete content and organizational editing before formatting. Please take note of the following items when proofreading spelling and grammar:

A. Percentage Accuracy and Computation Time of Algorithms on Different Datasets

a) Numeric and Text Datasets

For all three different datasets of mix datatype (numeric and text) as seen in Fig. 2, 3 and 4 mean accuracy percentage for both Random Forest and *k*-Nearest Neighbors are very close or almost same, whereas Naive Bayes show relatively lower performance. For Diabetes 130-US hospitals dataset in Fig. 3 Naïve Bayes performs very low, this suggests possible dependence among the attributes.

Fig. 5 shows the clear picture on mean % performance of mix datasets on each algorithm where RF and *k*NN perform equally well nevertheless Naïve Bayes shows lower performance.

Computation time for the Random Forest algorithm is highest for all datasets and least for Naïve Bayes. *K*-nearest neighbors took less time than random forest then more time than Naïve Bayes. With fix value of *k*, *k*-NN performs relatively faster than when using cross-validation to find optimal *k* value. This will be further discussed in next section.

TABLE II. Algorithm Mean Accuracy of Different Type of Datasets

Type of Dataset	Dataset Name	No. of Dataset Instances tested	No. of attributes	%Mean Accuracy			Computation Time (sec)		
				RF	K-NN	NB	RF	K-NN	NB
Numeric + Text	Online Retail Data Set	3000	8	94.7	94.7	90.6	598.37	190.80	3.08
Numeric + Text	Diabetes 130-US hospitals for years 1999-2008 Data Set	3000	52	79.19	78.9	31.43	854.97	114.815	2.23
Numeric + Text	Adult Data Set	3000	14	75.53	75.4	70.96	546.46	313.50	0.52
Numeric Only	Skin Segmentation Data Set	3000	4	100	100	100	196.78	111.77	0.39
Numeric Only	Pima Indians Diabetes Data Set	768	8	64.868	64.86	50.00	27.8	8.63	0.05
Numeric Only	Breast Cancer Data Set	699	10	78.55	77.97	88.99	36.51	4.34	0.06
Text Only	Nursery Data Set	3000	8	56.967	94.83	30.97	517.14	199.876	0.34
Text Only	News Aggregator Data Set (newscorp)	3000	7	100	100	100	375.84	261.12	3.15

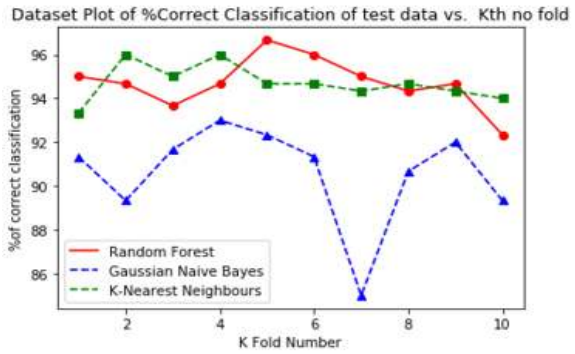

Fig. 2. Online retail data set (#541909) prediction accuracy with RF, *k*NN and NB classifiers.

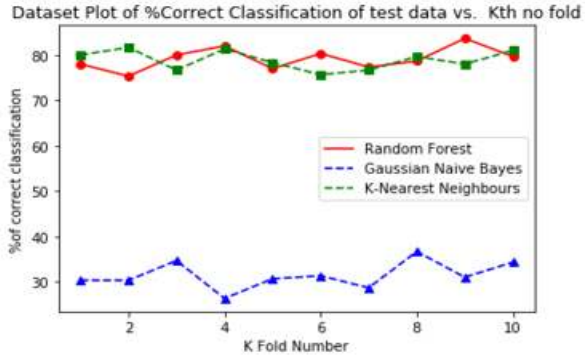

Fig. 3. Diabetes 130-US hospitals for years 1999-2008 Data Set (#100000) prediction accuracy with RF, *k*NN and NB classifiers.

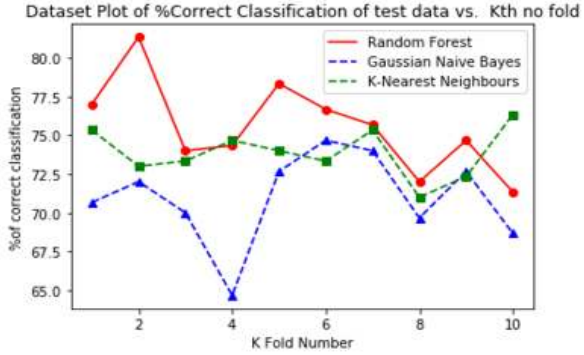

Fig. 4. Adult Data Set (#48842) prediction accuracy with RF, *k*NN and NB classifiers.

b) Numeric only: As shown in Fig. 6 for numeric only datasets all three behave identically for Skin Segmentation Dataset with 100% accuracy. Fig. 7 for Pima Indian Diabetes dataset illustrates the identical performance of RF and kNN in terms of accuracy whereas NB is performing lower possibly due to attributes in relation or dependency.

For Breast Cancer, RF and *k*NN have shown similar performances, whereas Naïve Bayes outperforms the other two, resulting in higher performances around 89% in Fig. 8. Fig. 9 showcases the identical performance of RF and *k*NN nonetheless for NB and its mix of equal, high and low performance for Skin Segmentation, Pima Indians Diabetes and Breast Cancer datasets, respectively.

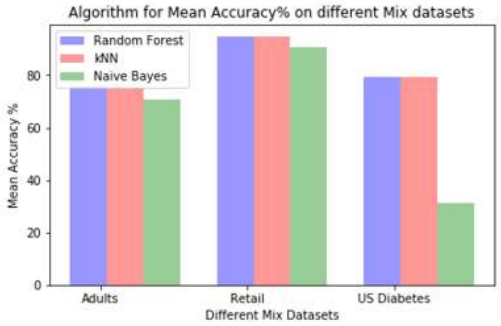

Fig. 5. Mean % accuracy of different mix datasets: Adults (#48842), Online Retail (#541909), UD Diabetes (#100000).

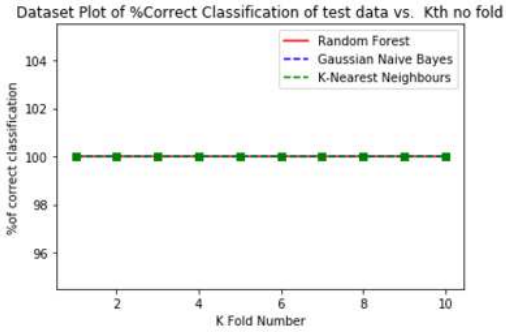

Fig. 6. Skin Segmentation Data Set (#245057) prediction accuracy with RF, *k*NN and NB classifiers.

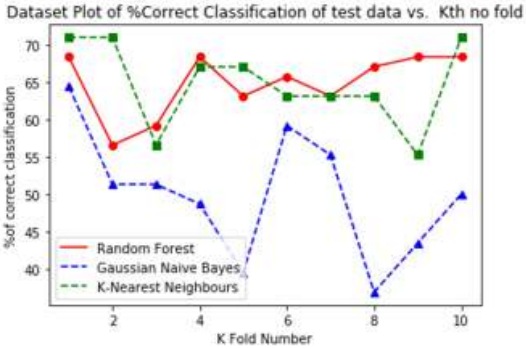

Fig. 7. Pima Indians Diabetes Data Set (#768) prediction accuracy with RF, *k*NN and NB classifiers.

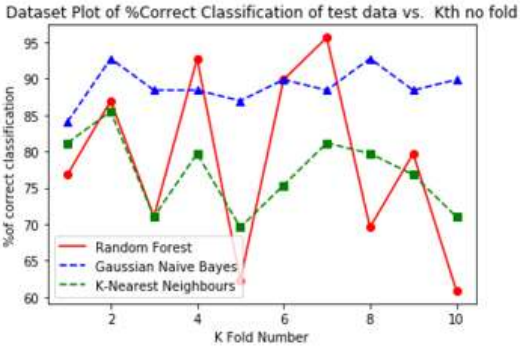

Fig. 8. Breast cancer data set (#699) prediction accuracy with RF, *k*NN and NB classifiers.

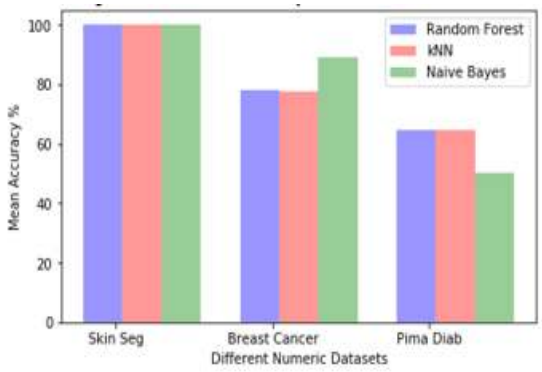

Fig. 9. Mean % accuracy of different numeric only datasets: Skin Segmentation (#245057) Breast Cancer (#699) and Pima Indians Diabetes (#768).

c) Text only

For Nursery data set in Fig. 10, K-Nearest Neighbors performance outperforms Random Forest and Naïve Bayes by giving mean a percentage accuracy of 94%. Naïve Bayes performs low with mean percentage accuracy around 31%, which indicates dependency among the dataset features or attributes. News Aggregator Data Set gives 100% mean accuracy with all three algorithms as seen in Fig. 11.

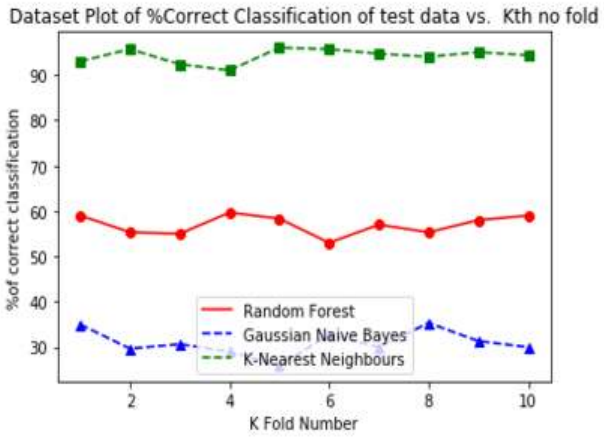

Fig. 10. Nursery data set (#12960) prediction accuracy with RF, *k*NN and NB classifiers.

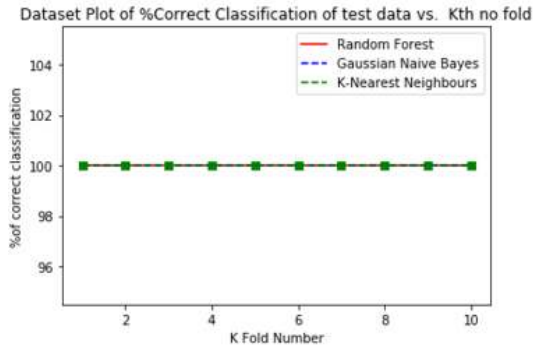

Fig. 11. News aggregator data set (newscorp.csv) (#417555) prediction accuracy with RF, *k*NN and NB classifiers.

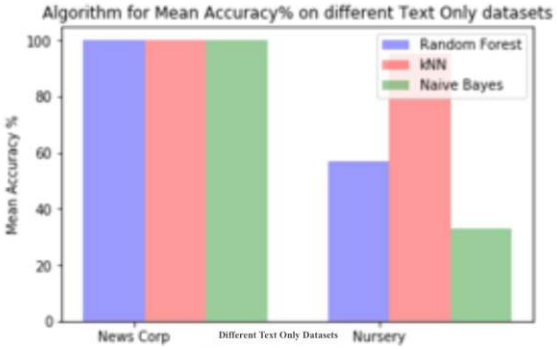

Fig. 12. Mean % accuracy of different text only datasets: News Aggregator Data Set (newscorp.csv) (#417555) and Nursery Data Set (#12960).

Fig. 12 shows that RF, *k*NN and NB perform similarly for News Aggregator Dataset but for Nursery Data Set *k*NN performs the best followed by RF then NB.

B. Optimal k for k-Nearest Neighbor

As seen in Table III, the cross-validating with different *k* values over sub set of data *k* =1 to 50 odd values that can give the suitable *k* value with highest % mean accuracy. Selecting a low fixed value of *k* =1 has the risk of over-fitting due to noise present in the training data set [20]. Consequently, we used the lowest odd value of *k*=3 which leads to less computation time nevertheless less mean accuracy compared to cross-validation.

TABLE III. COMPARING COMPUTATION TIME AND ACCURACY IN FINDING OPTIMAL *K* IN *K*NN OVER CROSS-VALIDATION ON A SUB SET OF DATA SET AGAINST FIX VALUES OF *K* LOW AS 3 AND HIGH AS LARGEST NUMBER SQUARE ROOT OF A NUMBER OF INSTANCES IN DATASET

Dataset used	**Using cross-validation**			**Fix Value $k = 3$**		**Fix value odd $k < \sqrt{n}$**		
	Computation time (sec)	**% Mean Accuracy**	**Optimal *k***	**Computation time (sec)**	**% Mean Accuracy**	***k***	**Computation time (sec)**	**% Mean Accuracy**
Mix: Adult Dataset (#3000)	335.31	75.3	19	95.49	72.07	53	93.042	75.53
Numeric: Pima Indian Dataset (#768)	100.81	64.47	35	3.51	53.29	25	3.86	62.89
Text: Nursery Dataset (#3000)	203.86	94.7	5	63.40	93.9	53	52.26	90.43

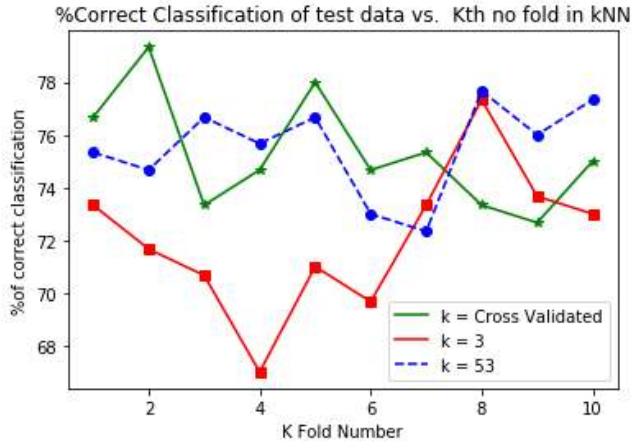

Fig. 13. Mean Accuracy % over two different values of k, k=3 and k=53 for mix dataset adult (#48842).

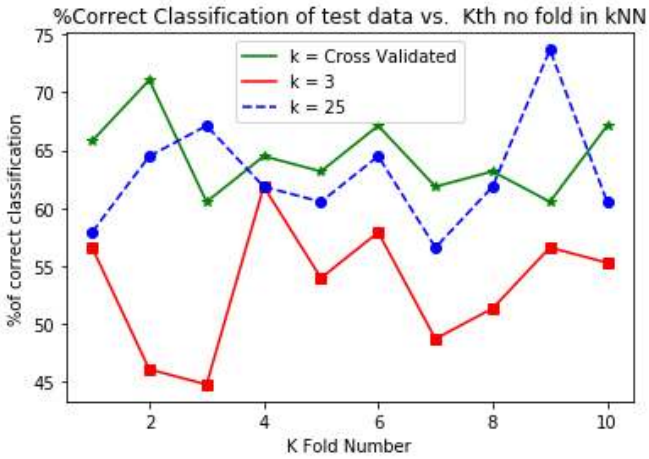

Fig. 14. Mean accuracy % over two different values of k, k=3 and k=25 for numeric only dataset pima indian diabetes (#768).

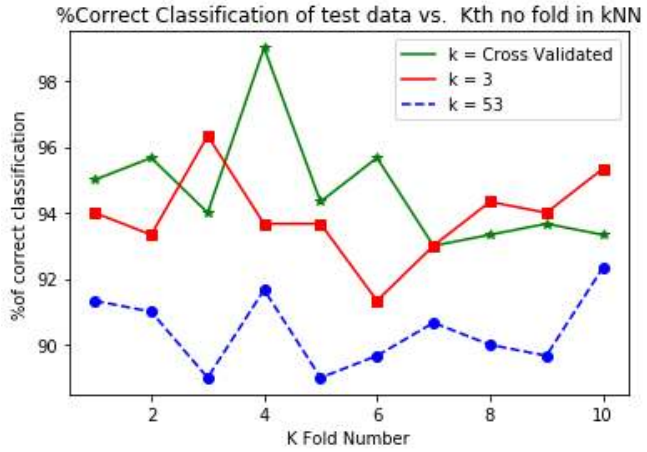

Fig. 15. Mean accuracy % over two different values of *k*, *k*=3 and *k*=53 for text only dataset nursery (#12960).

Fig. 13 and 14 shows taking k = 3 for Adult and Pima Indian dataset which gives the lowest performance nevertheless then again for Nursery dataset in Fig. 15 it gives better performance than the higher fixed value of k. When selecting a random high fixed value of k, using the general rule of thumb: $k < \sqrt{n}$ [20], where n is the number of instances in a dataset, and adult dataset in Fig. 13 shows marginal improvement whereas other two data sets have lower performance. Thus, choosing random low k values will lead faster computation with little lower accuracy that may be used for applications which need a quick prediction such as in disaster management. Also, choosing the higher odd value of k closest to the square root of the number of datasets also does not guarantee better accuracy. Depending on application requirement one can choose using a fixed k value or cross-validation to find optimal k.

C. Algorithm Performance on Chnaging the Number of Attributes

Following Table IV shows the behavior of all three algorithms with a change in number of attributes in terms of performance and computation time on Nursery Dataset with 1000 instances.

It can be clearly seen in Fig. 16 that random forest has no effect with increasing number of attributes on performance nonetheless the computation time increases.

For the *k*-Nearest Neighbors performance improves then remains constant over few more additions in attributes and lastly with full number of attributes its performance is highest at 93.3%.

Naïve Bayes takes the least time in computation overall and increasing attributes increases computation time insignificantly. Increasing number of attributes to 2 reduces performance drastically low as 0.2% and then goes up and down resulting 19% mean accuracy with complete set of attributes in Dataset.

D. Random Forest Performance with Different Number of Trees

Choosing number of trees for a given dataset is one of the important parameter optimization in random forest algorithm.

For this study, the number of trees used was 5, 10, 15, 20 and 25 on the Dataset.

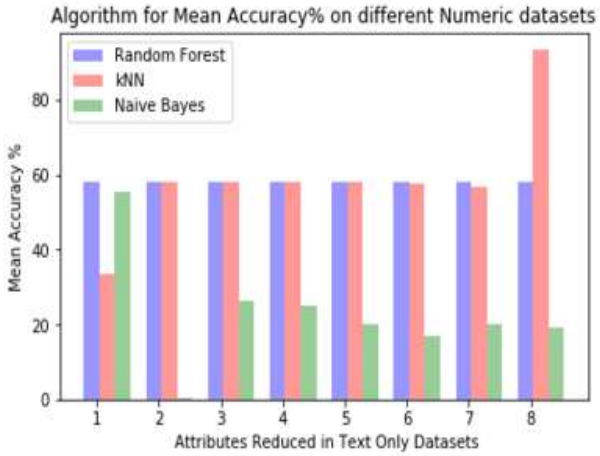

Fig. 16. Mean accuracy % over increasing number of attributes in nursery dataset (text only).

TABLE IV. EFFECT ON ACCURACY AND COMPUTATION TIME WITH CHANGE IN A NUMBER OF ATTRIBUTES IN NURSERY DATASET

No. of attributes	Computation Time (sec)			Accuracy (%)		
	Random Forest	*k*-Nearest Neighbors	Naïve Bayes	Random Forest	*k*-Nearest Neighbors	Naïve Bayes
1	18.897	3.191	0.096	57.9	33.3	55.3
2	18.456	4.007	0.0610	57.9	57.9	0.2
3	36.740	4.657	0.0723	57.9	57.9	26.2
4	38.237	5.543	0.080	57.9	57.9	20.2
5	39.472	5.876	0.094	57.9	57.7	17.1
6	36.744	6.533	0.106	57.9	56.6	20.0
7	42.526	7.217	0.124	57.9	93.3	19.0

TABLE V. SHOWS PERCENTAGE MEAN ACCURACY AND COMPUTATION TIME WITH DIFFERENT NUMBER OF TREES IN RANDOM FOREST ON ONLINE RETAIL DATASETS

No. of Trees	Computation Time (sec)	Accuracy (%)
5	588.75	94.7
10	1223.61	94.6
15	1766.75	94.7
20	2356.60	94.7
25	2994.56	94.7

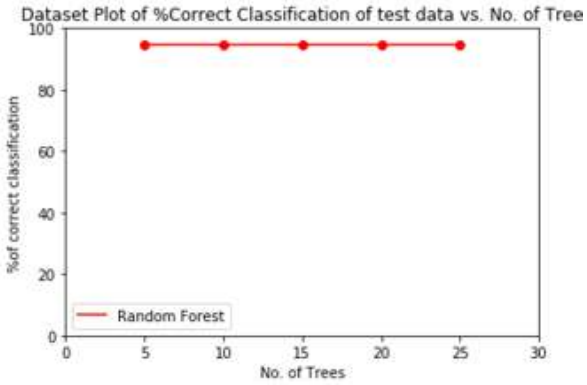

Fig. 17. The change in mean accuracy with change in number of trees for RF with Online Retail data set.

Table V and Fig. 17 clearly shows that increasing number of trees does not change mean accuracy significantly but computation time does increases greatly.

IV. DISCUSSION

This study has performed a comparative study of different types of big data sets with smaller data sets in Naïve Bayes, Random Forest, and *k*-Nearest Neighbor Classification Algorithms. The experimental results above show that RF and *k*NN perform significantly better than NB except in three datasets, Breast Cancer dataset where NB has the highest accuracy mean, on Skin segmentation and News Aggregator where all three algorithms have 100% mean accuracy. Thus, we can generalize that NB is the least performing, as this is possibly due to its independence assumption which is in accordance with [21] for all types of dataset. The RF and *k*NN perform similarly, as numeric only and text only datasets as seen in Table II except for nursery dataset where *k*NN outperforms RF significantly.

In terms of computation time, NB is the fastest of all, followed by *k*NN, and however, RF performs least. For datasets that require a quick prediction *k*NN can be the best choice. As seen in Table III using the optimal number of *k* effects the % accuracy is greatly [22]. When using the low fixed odd value there is high chance of overfitting and using the high fixed odd value less than $\sqrt{n}$ (n is number of data instances) does not guarantee better performance. The best possible way is to cross-validate and find optimal *k* to be used by the classifier [23].

It has been found that RF has no effect on changing the number of attributes as seen in experimental results, whereas % mean accuracy for *k*NN fluctuates from low to high performance and ends with high performance. For NB % mean accuracy fluctuates from high to low and ends with low performance showing variable dependency.

The performance of Random Forest algorithm increases with increasing number of trees and converges after some point. In this study, it has been clearly seen that the accuracy converging to 94.7% for N = 5 to 25 (N is number of trees at each node) on Online Retail datasets. In this case, large number of trees does not make a significant difference in performance. Large number of trees reduces the risk of overfitting and variance in the model [24] nonetheless it causes "curse of dimensionality" which makes model training inefficient. Using the default, results in large number of trees such as 500 performs well in many cases as seen in this study [23]. However, increasing the number of trees increases computation time which is the cost for large number of trees. Our study does not cover memory consumption in this case nonetheless it can be noted from [21] the large number of trees that consumes a lot of RAM space. Machine learning seems to be a great tool since it verifies some unexplained correlations in different attributes in any application [24]-[27].

V. CONCLUSION

This study has evaluated influence of three different data types (Text only, Numeric Only and Text + Numeric) on classifier performance (Random Forest, *k*-Nearest Neighbor (*k*NN) and Naïve Bayes (*NB*) algorithms). The classification problems in this study has been explored in terms of mean accuracy and the effects of varying algorithm parameters over different types of datasets. This content analysis has been examined through content examines at eight different datasets taken from UCI to train models for all three algorithms. These

datasets are of three different types: Text only, Numeric Only and Text + Numeric. This paper found that the best performing classifiers for a dataset such as a mix, text are only the numeric one's results. The results clearly show that Random forest and k-Nearest Neighbor (*k*NN) datasets behave identically. Naïve Bayes have shown significantly lower mean accuracy in few data sets indicating possible relation or dependency among dataset attributes as Naïve Bayes works with the assumption of attributes independence. In terms of computation time, random forest always takes more time and it increases further with an increase in number of attributes and numbers of trees to be built. Finding an optimal number of the tree for Random forest and k for *k*NN with cross-validation testing improves mean accuracy at the cost of computation time. Hence selecting a learning algorithm depends on the requirements of the problem application. Increasing the number of features undertaken by classifier also increases the feature space dimension causing "curse of dimensionality" and makes learning complicated with lower accuracy and higher computation time. This study could be further extended by using parametric methods to compare different algorithms on multiple data sets.

AUTHOR'S CONTRIBUTION

A.S. and M.N.H. conceived the study idea and developed the analysis plan. A.S. analyzed the data and wrote the initial paper. M.N.H. helped preparing the figures and tables, and in finalizing the manuscript. All authors read the manuscript.

APPENDIX

Algorithm 1: Main – Call different algorithm to predict and return mean accuracy prediction

```
N_FOLDS = 10
Load and prepare dataset
Set the parameters required by each algorithm
Create N_FOLDS random split of dataset into
train_set and test_set data
mean_acc= []: Array holding % Mean accuracy of RF,
kNN, NB
mean_acc=random_forest(train_set, test_set,
n_features, n_trees)
mean_acc.append(mean_accuracy)
mean_acc = k_nn(train_set, test_set,k)
mean_acc.append(mean_accuracy)
mean_acc=naïve_bayes(train_set, test_set)
mean_acc.append(mean_accuracy)
```

Algorithm 2: Random Forest

```
Calculate n_features = sqrt( total_no_of_features)
Set Constants
MAX_DEPTH = 10
MIN_SIZE = 1
SAMPLE_SIZE = 1.0
Create empty list trees
For i in range(NTREES):
  Create a random sub sample from dataset with
replacement of same size as dataset
  Create tree in sub sample:
    Create n_features number of features list from
dataset randomly.
    For each index in features
      For each row in sub sample
               Split the sub sample in groups
               Calculate gini to evaluate split and
               return best split point
        End For
    End For
    Split the dataset to create tree using best
split point for dataset sub sample
    Return this tree.
  Append this tree to trees list.
  End For
  Get prediction on test data with list of trees:
        For each row in test dataset:
               Make predictions with list of bagged
        trees and store number of predictions for
        each class list.
        End For
        Return Class with maximum number of votes in
prediction
        as a predicted target value.
  Calculate accuracy using predicted list of target
values and actual target values.
```

Algorithm 3: k- Nearest Neighbors

```
Calculate optimal k with least misclassification
error.
Create empty list predictions
For each t in testSet
         Find neighbors:
           Create empty list distances
           For each x in trainingSet
                dist= Euclidean distance measure
between x and test instance t
                distances.append(trainingSet[x],
        dist)
           End For
           Sort distances in ascending order.
           Create list neighbors by taking k subset
of training points from distances
            Return neighbors
        result = maximum voted class in the
neighbors
        predictions.append(result)
End For
Calculate accuracy using predicted list predictions
of target values and actual target values
```

Algorithm 4: Naïve Bayes

```
Summarize trainingSet:
  Separate trainingSet data by class
  Calculate mean and standard deviation for each
attribute
   Summarize for each class value
     Get predictions:
       predictions = []
         For i in range(len(testSet))
           Calculate result:
            Calculate Gaussian Probability Density
Function for given attributes in testSet[i]
             Calculate probability of the entire data
        instance belonging to the classes by
        multiplying probabilities of all the
        attribute values for testSet[i].
            Return the class with largest
probability as result.
          predictions.append(result)
        End For
        Return predictions
  Calculate accuracy using predicted list
predictions of target values and actual target
values.
```

REFERENCES

[1] I. H. Witten, E. Frank, M. A. Hall, C. J. Pal, "Data Mining: Practical Machine Learning Tools and Techniques". Morgan Kaufmann, 2016.

[2] V. M. Schonberger and K. Cukier. "Big Data: A Revolution that will Transform that will how we Live, Work, and Think" Boston USA, 2013.

[3] B. Heung, H. C. Ho, J. Zhang, A. Knudby, C. E. Bulmer and M. G. Schmidt, "An overview and comparison of machine-learning techniques for classification purposes in digital soil mapping". *Geoderma*, vol. 265, pp. 62 - 77, 2016.

[4] G. Dougherty (2013), "Pattern Recognition and Classification: An Introduction", Springer New York ISBN 978-1-4614-5323-9.

[5] T.T. Wong. Parametric methods for comparing the performance of two classification algorithms evaluated by k-fold cross validation on multiple data sets. Pattern Recogn. 65, pp. 97-107, 2017.

[6] S. S. Nikam. A Comparative Study of Classification Techniques in Data Mining Algorithms. Orient.J. Comp. Sci. and Technol; 8(1), April 2015

[7] S. B. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques". *Informatica*, Vol 31, pp 249-268, 2007

[8] S. Zhao, C. Rui and Y. Zhang, "MICkNN: multi-instance covering kNN algorithm," in *Tsinghua Science and Technology*, vol. 18, no. 4, pp. 360-368, August 2013.

[9] A. Danades, D. Pratama, D. Anggraini and D. Anggriani, "Comparison of accuracy level *k*-Nearest Neighbor algorithm and Support Vector Machine algorithm in classification water quality status," 2016 6th International Conference on System Engineering and Technology (ICSET), Bandung, 2016, pp. 137-141.

[10] V. Sucharita, S. Jyothi and P. V. Rao, "Comparison of machine learning algorithms for classification of Penaeid prawn species," *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, New Delhi, 2016, pp. 1610-1613.

[11] J. Chen et al., "A Parallel Random Forest Algorithm for Big Data in a Spark Cloud Computing Environment," in IEEE Transactions on Parallel and Distributed Systems, vol. 28, no. 4, pp. 919-933, April 1, 2017.

[12] L. Breiman. Random Forests. Machine Learning, 45 (1) (2001), pp. 5–32

[13] A. Liaw and M. Wiener. Classification and Regression by randomForest. *R News 2(3*), 18—22, 2003

[14] Ned, Horning. Random Forests: An algorithm for image classification and generation of continuous fields data sets. *International Conference on Geoinformatics for Spatial Infrastructure Development in Earth and Allied Sciences.* 2010.

[15] A. Wålinder. "Evaluation of logistic regression and random forest classification based on prediction accuracy and metadata analysis." (2014).

[16] UCI Machine Learning repository. Center for Machine Learning and Intelligent Systems.

[17] J. G. Moreno-Torres, J. A. Saez and F. Herrera, "Study on the Impact of Partition-Induced Dataset Shift on k-Fold Cross-Validation," in IEEE Transactions on Neural Networks and Learning Systems, vol. 23, no. 8, pp. 1304-1312, Aug. 2012.

[18] F. Nurwanto, I. Ardiyanto and S. Wibirama, "Light sport exercise detection based on smartwatch and smartphone using k-Nearest Neighbor and Dynamic Time Warping algorithm," 2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE), Yogyakarta, 2016, pp. 1-5.

[19] D. G. Huang, L. Guo, H. Y. Yang, X. P. Wei and B. Jin, "Chemical Medicine Classification Through Chemical Properties Analysis," in IEEE Access, vol. 5, pp. 1618-1623, 2017.

[20] Ming Leung, K.: 'k-nearest neighbor algorithm for classification', 2007. Available at http://cis.poly.edu/mleung/FRE7851/f07/k-NearestNeighbor.pdf

[21] Ensemble methods, Scikit-learn. http://scikit-learn.org/stable/modules/ensemble.html#forest

[22] M. T. Van, N. V. Tuan, T. T. Son, H. Le-Minh and A. Burton, "Weighted k-nearest neighbor model for indoor VLC positioning," in IET Communications, vol. 11, no. 6, pp. 864-871, 4 20 2017.

[23] Hassanat, A.B., Abbadi, M.A., Altarawneh, G.A., et al.: 'Solving the problem of the K parameter in the KNN classifier using an ensemble learning approach', Int. J. Comput. Sci. Inf. Sec., 2014, 12, (8), pp. 33–39

[24] Patel, J., Shah, S., Thakkar, P., & Kotecha, K. (2015). Predicting stock and stock price index movement using Trend Deterministic Data Preparation and machine learning techniques. Expert Systems with Applications, 42, 259–268.

[25] M. N. Halgamuge, "Machine Learning for Bioelectromagnetics: Prediction Model for Weak Radiofrequency Radiation Effect from Mobile Phone on Plants" International Journal of Advanced Computer Science and Applications (IJACSA), Accepted, November 2017.

[26] C. Wanigasooriya, M. N. Halgamuge, A. Mohamad, "The Analyzes of Anticancer Drug Sensitivity of Lung Cancer Cell Lines by Using Machine Learning Clustering Techniques", International Journal of Advanced Computer Science and Applications (IJACSA), Volume 8, No 9, September 2017.

[27] A. Gupta, A. Mohammad, A. Syed, and M. N. Halgamuge, "A Comparative Study of Classification Algorithms using Data Mining: Crime and Accidents in Denver City the USA", International Journal of Advanced Computer Science and Applications (IJACSA), Volume 7, Issue 7, pp 374 - 381, August 2016.