[1]Azza A. Nail
[2] Mohamed H. Ibrahim
[3] Mohamed H. Khafagy
[4] Mostafa R. Kaseb

Classification of Negative Databases

Journal of Electrical Systems

Abstract: - Due to the rapid growth and diversity of data, it has become essential to develop effective methods for maintaining data privacy when applying data mining algorithms. Large-scale data environments require secure methods that enable the classification and prediction of new inputs while preserving data confidentiality. This paper will propose farmwork for implementing classification algorithms based on the concept of negative databases (NDBs), aiming to perform binary and multi classification in secure environment. It will evaluate in terms of performance metrics, and prediction.

Keywords: Negative Databases (NDBs), Data Mining, Classification.

I. INTRODUCTION

Big data means large and complex datasets that are difficult to manage, process, or analyze using traditional methods. These datasets often derive from various sources such as social media, government data, sensors, and transactional data. The characteristics of big data are defined by "5Vs": volume, the generation of large amounts of data every second from millions of sources, potentially reaching terabytes (TB) or petabytes (PB) of data; velocity, the high speed at which data is generated and processed, data flows in real time; variety, the multiple types of data including structured, unstructured, and semi-structured formats such as post containing text, an image, and a location; veracity, represent the uncertainty or trustworthiness of the data, the quality and accuracy of the data must revise before analyzing it; and value, which extract the knowledge and benefits from data to help in decision making and improve performance. Additional V's such as variability Data changes over time and is not constant; visualization, the ability to present data in a visually understandable form; validity, accuracy, and truthfulness of data in relation to real world [1][2][3].

Data Mining is the process of extracting valuable information and knowledge from large and complex data. It consists of several stages, including data understanding, preparation, modelling, evaluation, and knowledge presentation. Data mining techniques are clustering, classification, and regression. It helps organizations to get knowledge to improve their profit. Data mining applications are in different fields such as banking, healthcare, agriculture, and education. The role of data mining analysis of the past and predicting the future. It integrates different techniques from statistics, database systems, and machine learning to analyze data [1].

Negative Databases (NDBs) represent a new type of data and a new technique for privacy-preserving, which derived from artificial immunes system, proposed by Esponda. NDB stores information that does not exist in its original database (DB). The universe(U) represents the binary string set, and DB denotes a set of binary strings. The size of DB is less than that of U−DB, so The NDB does not directly store U−DB. In consequence, NDB is the compression version of U−DB by introducing "∗" a wildcard symbol to match either "0" or "1". After the compression by introducing "*", the NDB size decreased. For example, DB is{0101},U−DB is{0000,0001,0010,0011,0100,0110,0111,1000,1001,1010,1011,1100,1101,1110,1111}, after compress U−DB ,NDB can be {1***, *0**, **1*, ***0}.Many algorithms, proposed for generating NDBs hard-to-reverse however, these algorithms are essentially focused on improving efficiency that mention in [18][19][20].

The remainder of this paper organized as follows: Section 2 outlines related works. Section 3 explains the methodology. Section 4 discusses experimental results. Section 5 concludes the paper and presents future work.

[1] *Corresponding author
[1,2] Information system Department, Faculty of Computers and Artificial Intelligence, Fayoum University, Fayoum, Egypt.
[3,4] Computer Science Department, Faculty of Computers and Artificial Intelligence, Fayoum University, Fayoum, Egypt.

Emails: [1]aam12@fayoum.edu.eg; [2] mhi11@fayoum.edu.eg ; [3] mhk00@fayoum.edu.eg; [4] Mrk00@fayoum.edu.eg

II. RELATED WORKS

Recent research has introduced various methods of big data classification algorithms [7]: In [8], Ravindran et al. introduced method called Data-Independent Reusable Projection (DIRP) the purpose of research to reduce the computational complexity and dimensionality reduction techniques used in classification of big data. The experiment depended on three benchmark datasets using the k-Nearest Neighbor (kNN)classifier. The results demonstrate that DIRP reduced computational complexity and runtime while maintaining a minimal loss of classification accuracy when compared with the full-dimensional data and the Principal Component Analysis (PCA) method. Also, in [9], Sleeman et al. introduced comprehensive methodology for handling imbalanced big data for multiclass and proposed efficient multi-class implementations sampling methods called SMOTE, finally, they introduced a clustering-based data partitioning this way able to leverage SMOTE. The experiment demonstrated that the proposed improvement for clustering-based also resulted in significant increases in predictive capability, making it more suitable for distributed environments. In [10]. Game et al. developed a new Divergence-based grey wolf optimization (DGWO) algorithm in health care. The developed approach consisted of three steps: (1) MapReduce framework; information was subjected to the MapReduce framework, where some operations were conducted to minimize the content of the data. However, MapReduce architecture utilized PCA to reduce dimensions present in the data. (2) SVM ;the minimal data was passed through the SVM to classifiers. The results from SVM were used, known as data conversion, leading to optimal rule production in the DT classifier. (3) Optimized Decision Tree; the converted data were passed to DT, where the categorization was performed using the DGWO algorithm. The experiment demonstrated that the scheme achieved higher classification accuracy but failed to apply the technique over a wide range of applications. In [11], García-Gil et al. proposed a novel approach which was named the DeTE_SD method depending on Decision Tree Ensemble with smart data for handling the imbalanced classification problem in big data areas. This method was primarily based on learning various decision trees using distributed quality data for the ensemble process. The experiment demonstrated that the quality of the data was achieved by merging Random Discretization, PCA, and clustering-based Random Oversampling to achieve various smart data versions of original data.

According to other studies that addressed Privacy-preserving classification based on Negative Databases (NDBS) technique: In [12], R. Liu et al. proposed classification and clustering algorithms in NDBs. The main method in this paper is calculating the Hamming distance between a binary string and proposed q-hidden algorithm to convert the original data into a negative database. KNN classification algorithm and a k-means clustering algorithm developed in NDBs Based on this method, respectively. The experimental results demonstrated the NDBs considered a novel privacy preserving technique in data mining. Also, in [13], H. Liao et al. proposed a privacy-preserving KNN classification algorithm based on the Euclidean distance calculation formula on a negative database. The algorithm proposed in this research used the K-hidden algorithm to convert the original data into a negative database. It applied the Euclidean distance formula to calculate the distance between the negative training data and the test data to gain the result of classification. The experimental results achieved high classification accuracy and indicated that the proposed algorithm performed classification analysis while protecting data privacy. In [14], Zhao et al. proposed a method for estimating Euclidean distances from the negative databases. The authors also proposed QK-hidden algorithm to convert the original data into a negative database and controlling its generation. Based on this method, they proposed a privacy-preserving k-means algorithm and kNN classification algorithm. The work in this paper analyzed the efficiency and privacy of the proposed algorithms. The experimental results showed that the proposed algorithms achieved better performance in clustering and classification.

Based on the work presented in the relevant studies that were reviewed. Privacy-preserving classification methods rely on encryption typically require high computational cost, perturbation-based methods are often less security and suffer from low accuracy, and secure multi-party computing-based methods usually need high communication cost. Therefore, it is important to introduce new techniques to privacy-preserving classification tasks [17]. This proposed paper will develop classification algorithms based on negative databases. The main contributions of this paper include the following:

- Preparing original datasets for binary and multiclass classification.
- Generating NDBs from original datasets.
- Labeling NDBs.
- Developing classification models based on NDBs.
- Evaluating the performance of classification models based on NDBs.
- predicting new data on NDBs.

III. METHODOLOGY

The proposed Framework consists of four phases: (1) data preparation and preprocessing, (2) negative databases generation, (3) negative databases labeling and (4) classification of negative databases, as shown in Fig.1.

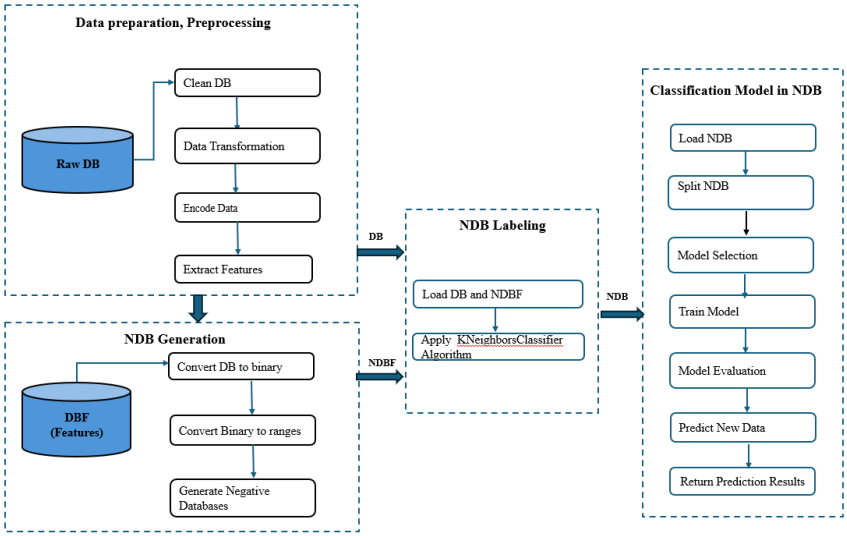

Fig.1. The Proposed Framework of Classification NDB.

A. Data Preparation and Preprocessing

This phase begins with downloading datasets from sources such as Kaggle or the UCI Machine Learning Repository. The first step, preparation, follows the data collection stage and includes cleaning, transforming, and organizing raw data into a structured format. This process ensures data accuracy and removes irrelevant columns or information that are not important for the classification task. The second step, preprocessing, involves several operations such as data transformation to convert floating-point numbers into integers, string encoding, and separating features and the target variable, with independent variables (features) and the dependent variable (target) prepared individually for modeling.

B. Negative Databases Generation

This is the second phase of the proposed framework, which involves generating the complement set from the original datasets (DB) and compressing it to minimize its size to create the Negative Database (NDB). The first step, convert each record in DB to binary representation; this step is presented in Algorithm (1):

Algorithm 1 Convert DB to Binary values
Input: *DBF* → original data features.
Output: *Binary_values* → list of binary representation.

1. Read *DBF*.
2. Delete any rows in the *DBF* that contain null values.
3. Encode *DBF* for string data.
4. Create an empty dictionary *max_lengths* to store the maximum binary length for each column.
5. For each col in the *DBF*
   a. Convert all values in *col* to integers.
   b. Convert each integer to a binary string.
   c. Find the maximum length of the binary strings in that column → *max_len.*
   d. Store *max_len* in *max_lengths[col].*
   e. padding all binary strings in the column starting with zeros to become *max_len.*
6. For each *row* in the *DBF*
   a. Merge the binary strings for all columns into a single string.
   b. Add this string to Binary_values.
7. Return Binary_values.

The second step, convert each binary representation to binary ranges; this step is presented in Algorithm (2).

Algorithm 2 Generate Binary Ranges
Input: *Binary_Values*: a list of binary strings (sorted in ascending order).
Output: *List_Ranges*: each range is [start, end].

1. Initialize an empty *list_Ranges.*
2. Set the *start* to a sequence of zeros (length (*Binary_values*)).
3. For each *bin* in *Binary_values*
   a. Decrement *bin* by 1

b. Assign *bin* to *end*.
c. Add range [*start*, *end*] to *list_Ranges*.
d. Increment *bin* by 1
e. Assign *bin to start*.
4. Set *end* to a sequence of ones (length (*Binary_values)).*
a. Add the last range [*start, end*] to *List_Ranges*.
5. Return *list_Ranges.*

The third step, Generate Negative Databases (NDBs) Algorithm; this step is presented in Algorithm (3).

Algorithm 3 Generate Negative Databases
Input: *Start* → start of range, End → end of range
Output: *NDBF* → list of compressed binary numbers using * for variable bits without target column.

1. Initialize *NDBF, Subinterval = False.*
2. *Assign State* to Compare (*Start*, *End).*
3. If *State* == 1
a. Return None.
4. Else If *State* == 0
a. Append *Start* to *NDBF*.
b. Return *NDBF*.
5. Else
a. If *Start* is odd
1. Append *Start* to *NDBF*.
2. Increment *Start* by 1.
3. If Compare (*Start*, *End)==0*
i. Append *Start* to *NDBF*.
ii. Return *NDBF.*
6. Convert *Start* to a list *Compress_Num.*
7. Initialize *index i*= len(*Compress_Num*) – 1.
8. Loop *i* >= 0
a. If *Compress_Num*[i] == '0'
1. Assign *Compress_Num*[i] to '*'.
2. Set *Temp* = *Compress_Num* with all '*' replaced by '1'.
3. State = Compare (*Temp*, *End*)
4. If State == 0
i. Append *Compress_Num* to *NDBF*.
ii. Return *NDBF*
5. Else If *State* ==1
i. break loop.
b. Else if *Compress_Num*[i] == '1'
1. If not *Subinterval*
i. Append current *Compress_Num* to *NDBF*.
2. Assign Compressed_Num[i] to '*'.
3. Adjust next higher bit if needed; update Subinterval.
4. Repeat steps from 2 to 5 in previous step a.
c. Decrement *i*.
9. While i < len(*Compress_Num*)
a. If *Compress_Num*[i] == '*':
1. Assign *Compress_Num*[i] to '0'.
2. Set *Temp* = *Compress_Num* with all '*' replaced by '1'.
3. *State* = Compare (*Temp*, *End*)
4. If *State* == 0
i. Append *Compress_Num* to *NDBF*.
ii. Return *NDBF*.
5. Else If State ==-1
i. Append *Compress_Num* to *NDBF.*
iii. Assign *Compress_Num* [i] to '1'.
b. Increment i.
10. Return NDBF.

C. Negative Databases Labeling

The third phase of the proposed model aims to add a target column to the negative database (NDB) after it has been generated from the original data (DB). This step is necessary because the negative database contains only feature columns, making it unsuitable for direct use in classification tasks. To achieve this, the NDB is employed as the test dataset, while the DB serves as the training dataset. The process is implemented using the K-Nearest Neighbors Classifier algorithm, which relies on the Euclidean distance to identify the closest instance in the test data and assign the corresponding class label accordingly. This step is presented in Algorithm (4).

Algorithm 4: Negative Databases Labeling
Input: *DB* → original dataset, *NDBF* → negative databases feature values to without target column, *k* → number of nearest neighbors , *target_column* → name of the predict column
Output: *Label* → predicted value for *NDBF*[target_column]

1. Read *DB* and *NDBF*.
2. *Set K=3.*
3. Define *Target* and *Feature* Columns of DB.
4. Split Training Data *DB*
   a. Set X_train ← DB[feature_cols].
   b. Set Y_train ← DB[target_column].
5. Create *knn* ← KNeighborsClassifier(K).
6. *knn*.fit(*X_train*, Y_*train*).
7. Extract *Feature* columns from new data
   a. Extract *Feature* columns from new data: X_new ←*NDBF* [feature_cols].
8. Compute *Label* ← knn.predict(X_new).
9. Add *Label* to *NDB*.

D. Classification of Negative Databases

The first step of last phase begins with negative database preprocessing, where the dataset is loaded and any missing or null values are removed. Next, the dataset is divided by separating the features and the target and split it into test, training and validation sets at 20%, 64%, and 16%, respectively. In the model selection step, an appropriate classification algorithm is chosen, such as logistic regression, K-nearest neighbors, decision tree, random forest or Gradient Boosting, and initialized with the required hyperparameters. This is followed by hyperparameter tuning, during which the model is trained and evaluated adjusting the parameters to optimal performance. The final model is trained on the complete training set, and its performance is evaluated on the test set by predicting labels and computing evaluation metrics. In the second step, the trained model is applied to new data, which is preprocessed in the same manner as the training data before predicting its label. Finally, the results are compiled and returned, including the trained model, predictions on the test data, and predictions for the new data. This step is presented in Algorithm (5).

Algorithm 5 Classification of Negative Databases
Input: *NDB*, *new_point*
Output: *Classification report, Label*

1. Read *NDB*
2. Split *NDB*
   a. Separate features *X* and labels *Y*
   b. divided the dataset
   c. X train, Y train
   d. X val, Y val
   e. X test, Y test
3. Model Selection
   a. Choose a classification algorithm.
   b. Initialize the model with required hyperparameters.
4. Hyperparameter Tuning
   a. Train the model on *X_train, Y_train.*
   b. Evaluate on *X val, Y val.*
5. Train Model
   a. Fit the final model using the training data (*X train, Y train*).
6. Evaluate Model on Test
   a. Predict labels on *X test*
   b. Compute evaluation metrics
7. Make Predictions on New Data
   a. Preprocess *new point* the same way as training data.
   b. Predict *label*
8. Return *label*, *Classification report*.

IV. EXPERIMENTAL RESULTS

This part is the classification evaluations of negative datasets, and experiments are conducted using binary and multi-class datasets. The experimental binary dataset comes from two original datasets: (1) The Breast Cancer Wisconsin Diagnostic (BCWD) Dataset is one of the most widely used datasets in medical research and data mining for binary classification problems. The dataset is designed to classify tumors into malignant or benign categories based on features. The dataset contains 569 patient instances, each described by 30 numerical features that capture various characteristics, along with one target label indicating the diagnosis. Importantly, the dataset is clean with no missing value, making it suitable for direct use in data mining experiments. In terms of class distribution, it consists of 37.3% malignant cases and 62.7% benign cases, as shown in Table 1, (2) The bank dataset records the

results of direct marketing campaigns from a Portuguese banking institution, where phone calls were made to clients and various socio-economic and contact features were captured. The dataset contains a set of 11162 records under 73 attributes; The dataset is designed to classify deposit into yes or no categories based on features.

The experimental multiclass dataset was derived from two original datasets: (1) The Iris flower dataset is a multivariate dataset. The dataset consists of 50 samples from each of 3 species of Iris and 4 features were measured from each sample. The dataset contains a set of 150 records under 5 attributes. It became a typical test case for many classification techniques in data mining. As shown in table 1, (2) The Crop Recommendation Dataset from Kaggle is designed to support research in precision agriculture by recommending suitable crops based on conditions of soil and environmental. Each record corresponds to a specific set of conditions and the most suitable crop label. The dataset includes more than 20 crop types, such as rice, maize, banana, apple, and coffee. It is well-structured, clean, and suitable for supervised classification tasks. Researchers can use it to develop predictive models that help stakeholders to select suitable crops based on weather patterns and soil quality. The dataset contains a set of 2200 records under 8 attributes, as shown in Table 1.

Table 1. Datasets Description.

Type of Classification	Original Datasets	Target Column	No. Features	No. instances (DB)	No. instances (NDB)
Binary	BCWD	(0,1)	30	569	273575
	Bank Dataset	(0,1)	73	11162	385485
Multiclass	The Iris flower	(0,1,2)	4	150	1898
	Crops	(0,…,20)	8	2200	167009

For the WDBC, Bank, Crops and Iris datasets, the classification Performance metrics is measured using the standard following formula, as shown in Table 2.

Table 2. Performance metrics formula.

Metric	Calculation
Accuracy	$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$
Recall	$\frac{TP}{TP+FN}$ True Positives (True Positives + False Negatives)
Precision	$\frac{TP}{TP+FP}$ True Positives (True Positives + False Positives)
F-score	$\frac{2 \cdot (Precision * Recall)}{Precision + Recall}$

The experiments detailed in this proposed work are conducted in a high-performance environment with powerful specifications. The hardware platform used a 12th generation Intel Core i5-12500H processor, operating at 2500 MHz, featuring a 12-core architecture with 16 logical processors, 16 GB of main memory and accelerated by a 4 GB graphics processing unit (GPU) to facilitate efficient classification results. The software used PySpark and PyCharm IDE.

A. Classification Models in Traditional Environment

The experiment is applied classification algorithms in the traditional environment using negative binary class datasets (BCWD-NDB, Bank-NDB), as shown in table. 3,4 and negative multi-class datasets (Iris-NDB, Crops-NDB), as shown in table. 5,6. The results are compared with previous studies.

Table 3. Classification Models Results in Traditional Environment in BCWD-NDB

Model	Accuracy(V)	Accuracy(T)	Precision	Recall	F-score
Logistic Regression	0.981	0.982	0.98	0.98	0.98
KNN	0.996	0.996	1.00	1.00	1.00
Decision Tree	0.999	0.999	1.00	1.00	1.00
Random Forest	0.999	0.999	1.00	1.00	1.00
Gradient Boosting	0.998	0.999	1.00	1.00	1.00

Table 4. Classification Models Results in Traditional Environment in Bank-NDB

Model	Accuracy(V)	Accuracy(T)	Precision	Recall	F-score
Logistic Regression	0.842	0.842	0.85	0.84	0.84
KNN	0.983	0.983	0.98	0.98	0.98
Decision Tree	0.981	0.981	0.98	0.98	0.98
Random Forest	0.974	0.974	0.97	0.97	0.97
Gradient Boosting	0.908	0.907	0.91	0.91	0.91

In BCWD-NDB, the performance of various classification models was evaluated using multiple metrics, including accuracy on the validation and test, precision, recall, and F1-score. Decision Tree achieved the highest performance, with perfect scores of 1.0 for precision, recall, and F1-score, and slightly higher accuracy on both validation 0.9994 and test 0.9992 sets. Random Forest also performed very well, reaching accuracy of 0.9990 and 0.9991 on the validation and test, respectively. Gradient Boosting showed strong results, with accuracy 0.9986 and precision, recall, and F1-scores around 1.0. KNN showed strong results, with accuracy close to 0.9956 and precision, recall, and F1-scores around 1.0. Logistic Regression demonstrated slightly lower performance compared to the others, with accuracy of 0.9812–0.9822 and corresponding precision, recall, and F1-scores around 0.98, as shown in Fig 2. In Bank-NDB, KNN and decision tree achieved the highest performance, with high scores of 0.98 for precision, recall, and F1-score, and higher accuracy on both validation and test around 0.98. Random Forest also performed very well, reaching accuracy on the validation and test around 0.97. Gradient Boosting showed strong results, with accuracy 0.91 and precision, recall, and F1-scores around 0.91. Logistic Regression demonstrated lower performance compared to the others, with accuracy of 0.84 and corresponding precision, recall, and F1-scores around 0.84, as shown Fig 2.

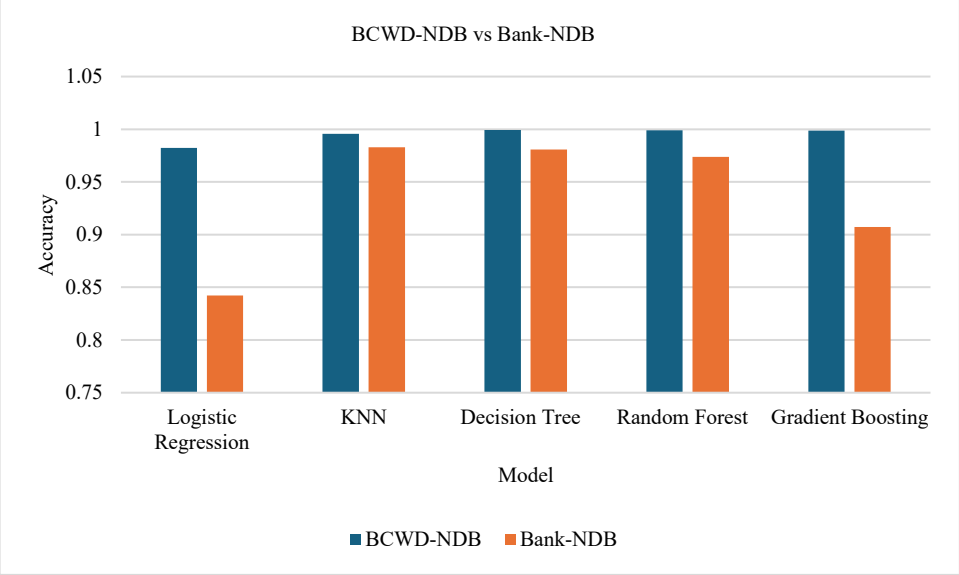

Fig. 2. Accuracy Comparison between Bank-NDB and BCWD-NDB.

Table 5. Classification Models Results in Traditional Environment in Iris-NDB.

Model	Accuracy(V)	Accuracy(T)	Precision	Recall	F-score
Logistic Regression	0.959	0.956	0.96	0.96	0.96
KNN	0.968	0.975	0.98	0.98	0.98
Decision Tree	0.976	0.980	0.98	0.98	0.98
Random Forest	0.980	0.982	0.98	0.98	0.98
Gradient Boosting	0.974	0.984	0.98	0.98	0.98

Table 6. Classification Models Results in Traditional Environment in Crops-NDB.

Model	Accuracy(V)	Accuracy(T)	Precision	Recall	F-score
Logistic Regression	0.806	0.804	0.79	0.81	0.79
KNN	0.990	0.990	0.99	0.99	0.98

Decision Tree	0.997	0.996	0.99	0.99	0.99
Random Forest	0.997	0.996	0.97	0.97	0.97
Gradient Boosting	0.993	0.993	0.99	0.99	0.99

In Iris-NDB, Gradient Boosting, Random Forest and Decision Tree achieved the highest performance with high scores of 0.98 for precision, recall, and F1-score, and higher accuracy on both validation and test around 0.98. KNN also performed very well, reaching accuracy on the validation and test around 0.975. Gradient Boosting showed strong results, with accuracy 0.956 and precision, recall, and F1-scores around 0.96. Logistic Regression showed well results, with accuracy close to 0.956 and precision, recall, and F1-scores around 0.96, as shown in Fig 3.In Crops-NDB, Decision Tree and Random Forest also achieved the highest performance, with high scores of 0.99 ,0.97 for precision, recall, and F1-score respectively, and higher accuracy on both validation and test around 0.996. KNN also performed very well, reaching accuracy on the validation and test around 0.990 and 0.99,0.99,0.98 for precision, recall, and F1-score. Gradient Boosting showed strong results, with accuracy 0.993 for validation, test, and precision, recall, and F1-scores around 0.99. Logistic Regression demonstrated lower performance compared to the others, with accuracy of 0.806 for validation and 0.804 for test and corresponding precision, recall, and F1-scores around 0.79,0.81,0.79, as shown in Fig 3.

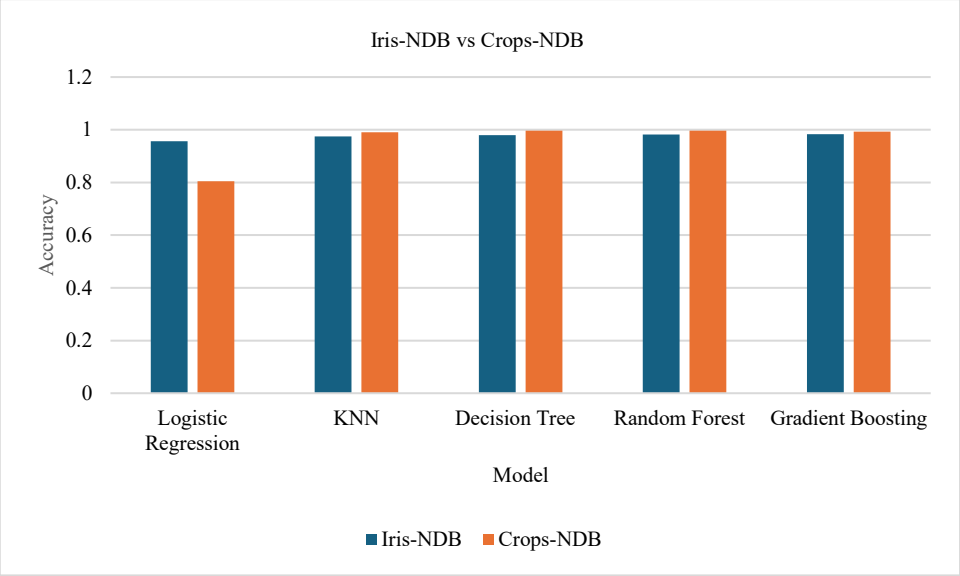

Fig. 3. Accuracy Comparison between Iris-NDB and Crops-NDB.

B. Classification Models with Pyspark

The experiment is applied classification algorithms in the Pyspark environment using negative binary class datasets (BCWD-NDB, Bank-NDB), as shown in table 7,8 And a negative multi-class dataset (Iris-NDB, Crops-NDB), as shown in Table 9,10.

Table 7. Classification Models Results with Pyspark in BCWD-NDB.

Model	Accuracy(V)	Accuracy(T)	Precision	Recall	F1-Score
Logistic Regression	0.982	0.983	0.98	0.98	0.98
Decision Tree	0.989	0.989	0.99	0.99	0.99
Random Forest	0.988	0.988	0.99	0.99	0.99
Gradient-Boosted Trees (GBT)	0.999	0.999	1.0	1.0	1.0

Table 8. Classification Models Results with Pyspark in Bank-NDB

Model	Accuracy(V)	Accuracy(T)	Precision	Recall	F-score
Logistic Regression	0.847	0.843	0.85	0.84	0.84
Decision Tree	0.938	0.937	0.94	0.94	0.94
Random Forest	0.933	0.933	0.93	0.93	0.93
Gradient-Boosted Trees (GBT)	0.953	0.953	0.95	0.95	0.95

In BCWD-NDB, GBT achieved the highest results with 0.999 validation accuracy, 0.999 test accuracy, along with perfect precision, recall, and F1-score of 1. Decision Tree also performed well, showing 0.989 validation accuracy, 0.989 test accuracy, and precision, recall, and F1-score of 0.99. Random Forest achieved slightly lower performance, with accuracy around 0.988 validation and 0.988 test, and precision, recall, and F1-score of 0.99. Logistic Regression showed the lowest results among the models, with validation and test accuracy of 0.982 and 0.983, and precision, recall and F1-score of 0.98, as shown as Fig 4. Also, In Bank-NDB, GBT achieved the highest results, with accuracy of 0.953 on the validation set and 0.953 on the test set, and precision, recall, and F1-score of 0.95. Decision Tree also performed well, showing accuracy of 0.937 validation and 0.937 test and corresponding metrics close to 0.94. Random Forest achieved lower performance, with accuracy around 0.933 validation and 0.933. Logistic Regression showed the lowest results among the models, with validation and test accuracy of 0.847 and 0.843, as shown as Fig 4.

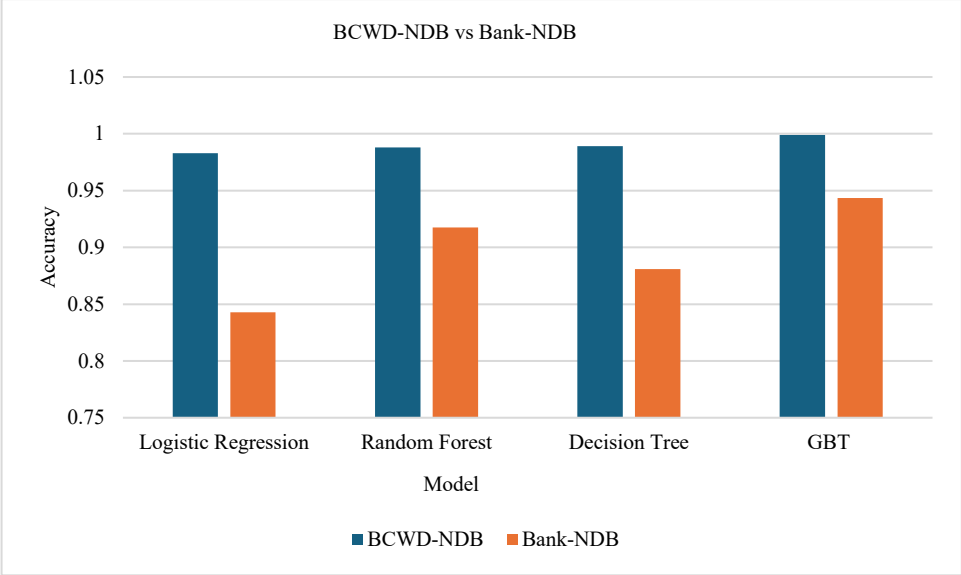

Fig. 4. Accuracy Comparison between Bank-NDB and BCWD-NDB on Pyspark ML.

Table 9. Classification Models Results with Pyspark in Iris-NDB.

Model	Accuracy(V)	Accuracy(T)	Precision	Recall	F1-Score
Logistic Regression	0.959	0.953	0.95	0.95	0.95
Decision Tree	0.967	0.972	0.97	0.97	0.97
Random Forest	0.969	0.970	0.97	0.97	0.97

Table 10. Classification Models Results with Pyspark in Crops-NDB.

Model	Accuracy(V)	Accuracy(T)	Precision	Recall	F-score
Logistic Regression	0.811	0.809	0.79	0.81	0.79
Decision Tree	0.977	0.976	0.977	0.976	0.976
Random Forest	0.987	0.987	0.987	0.987	0.987

In Iris-NDB, Decision Tree and Random Forest achieved the highest results with accuracy 0.97 on validation, test, and precision, recall, and F1-score of 0.97. Logistic Regression also performed well, showing accuracy of 0.959 Validation and 0.9818 Test, and precision, recall, and F1-score of 0.97 0.95, as shown as Fig 5. Also, In Crops-NDB, Random Forest achieved the highest results, with accuracy of 0.987 for validation and test, along with perfect precision, recall, and F1-score of 0.98. Decision Tree also performed well, showing accuracy of 0.977 validation and 0.976 test and corresponding metrics close to 0.97. Logistic Regression showed the lowest results among the models, with validation and test accuracy of 0.811 and 0.809, and precision, recall, and F1-score of 0.79,0.81,0.79 respectively, as shown as Fig 5.

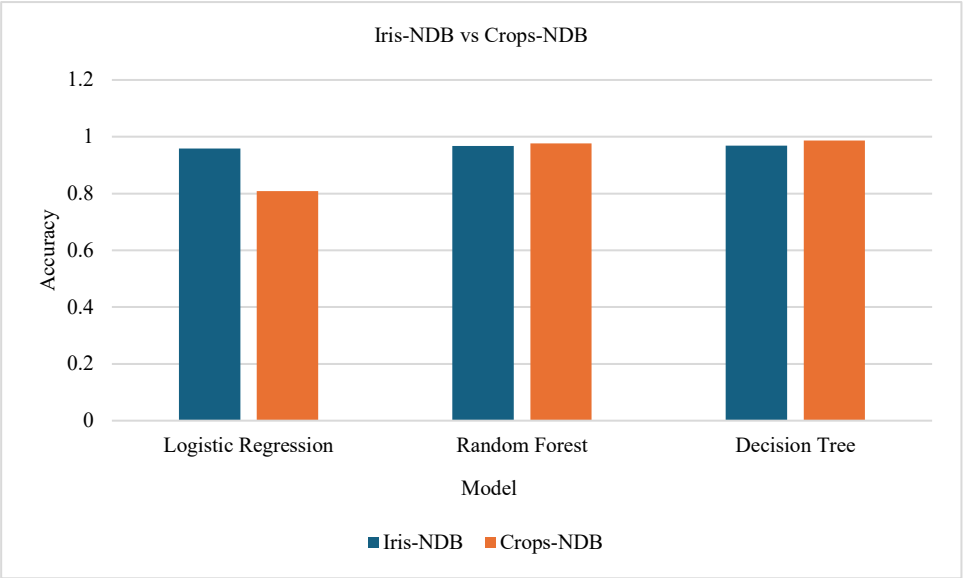

Fig. 5. Accuracy Comparison between Iris-NDB and Crops-NDB on Pyspark ML.

V. CONCLUSION

Based on this research, the proposed framework was introduced by four phases: (1) data preparation and preprocessing, (2) negative databases generation, (3) negative databases labeling and (4) classification of negative databases. The experiments compared negative binary and multi class datasets with traditional and big data environments. The results were evaluated and demonstrated the efficiency of negative databases in classification. The best algorithm was selected for predicting new input. Future work will be applying the concept of negative databases to neural networks.

ACKNOWLEDGMENT

I extend my sincere gratitude to the supervisors for their support, guidance, expertise, and constructive feedback, which played a vital role in the completion of this work.

REFERENCES

[1] Tyagi, Amit Kumar. "Machine learning with big data." Machine Learning with Big Data (March 20, 2019). Proceedings of International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM), Amity University Rajasthan, Jaipur-India. 2019.

[2] Salkuti, Surender Reddy. "A survey of big data and machine learning." *International Journal of Electrical & Computer Engineering (2088-8708)* 10.1 (2020).

[3] Boyapati, Sumati, et al. "Big data approach for medical data classification: A review study." 2020 3rd international conference on intelligent sustainable systems (ICISS). IEEE, 2020.

[4] Xing, Wenchao, and Yilin Bei. "Medical health big data classification based on KNN classification algorithm." Ieee Access 8: 28808-28819,2019.

[5] Mapca, B., et al. "Efficient privacy preservation of big data for accurate data mining-ScienceDirect." Information Sciences 527.3 (2020): 420-443.

[6] Khan, Saira, et al. "Clustering based privacy preserving of big data using fuzzification and anonymization operation." arXiv preprint arXiv:2001.01491 (2020).

[7] Abdalla, Hemn Barzan, and Belal Abuhaija. "Comprehensive analysis of various big data classification techniques: A challenging overview." Journal of Information & Knowledge Management 22.01: 2250083,2023.

[8] Ravindran, Siddharth, and G. Aghila. "A data-independent reusable projection (DIRP) technique for dimension reduction in big data classification using k-nearest neighbor (k-NN)." *National Academy Science Letters* 43.1 (2020): 13-21.

[9] Sleeman IV, William C., and Bartosz Krawczyk. "Multi-class imbalanced big data classification on spark." Knowledge-Based Systems 212 (2021): 106598.

[10] Game, Pravin S., Vinod Vaze, and M. Emmanuel. "Optimized Decision tree rules using divergence based grey wolf optimization for big data classification in health care." *Evolutionary Intelligence* 15.2 (2022): 971-987.

[11] García-Gil, Diego, et al. "Smart data driven decision trees ensemble methodology for imbalanced big data." *Cognitive Computation* 16.4 (2024): 1572-1588.

[12] Liu, Ran, Wenjian Luo, and Lihua Yue. "Classifying and clustering in negative databases." *Frontiers of Computer Science* 7.6 (2013): 864-874.

[13] Liao, Hucheng, et al. "Privacy-protected kNN classification algorithm based on negative database." The International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery. Cham: Springer International Publishing, 2019.

[14] Zhao, Dongdong, et al. "K-means clustering and kNN classification based on negative databases." *Applied soft computing* 110 (2021): 107732.

[15] F. Esponda, S. Forrest, and P. Helman, "Enhancing privacy through negative representations of data," Technical Report, DTIC Document, A667894, 2004.

[16] Esponda, Fernando, et al. "Protecting data privacy through hard-to-reverse negative databases." *International Journal of Information Security* 6.6 (2007): 403-415.

[17] Zhao, Dongdong, et al. "A fine-grained algorithm for generating hard-toreverse negative databases." *2015 International Workshop on Artificial Immune Systems (AIS)*. IEEE, 2015.