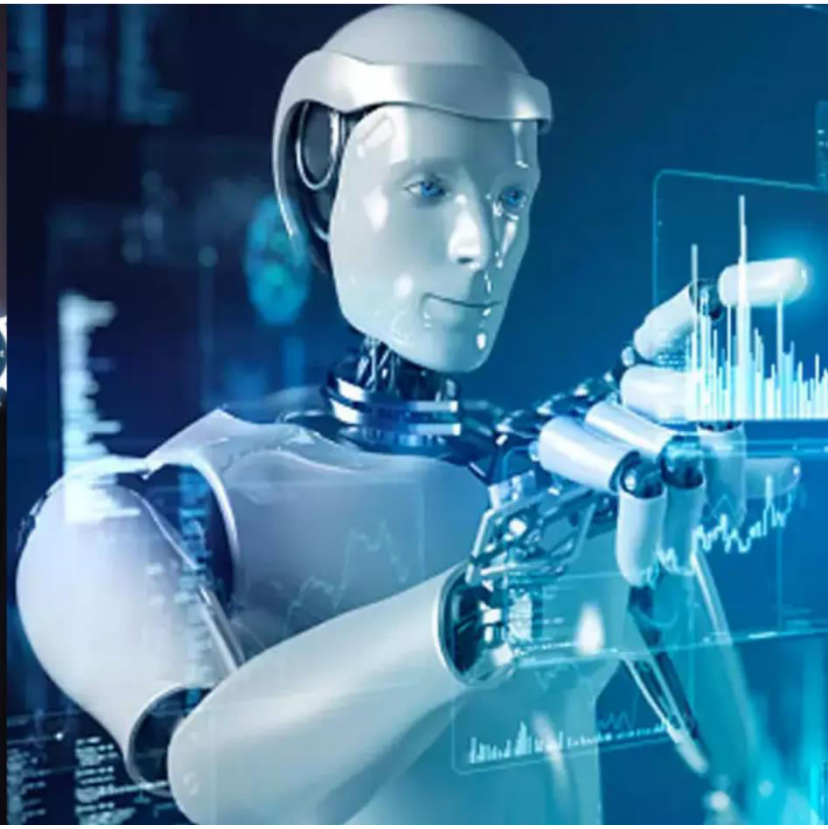
IJIRSET
International Journal of Innovative Research in Science | Engineering | Technology

ISSN(O): 2319-8753
ISSN(P): 2347-6710

International Journal of Innovative Research in Science Engineering and Technology (IJIRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.699

Volume 14, Issue 3, March 2025

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 3, March 2025

|DOI: 10.15680/IJIRSET.2025.1403383|

Algorithmic Archaeology: Unearthing Machine Learning by Building from Scratch

Ms. Sweety Patel [1], Mr, Nitin Pal[1] , Purvang Dave[2] , Aniket Priyadarshi[2], Anandkumar[2] , Akshay Kumar[2]

Assistant Professor, Department of Computer Science and Engineering, Parul Institute of Technology, Vadodara, Gujarat, India. [1]

B.Tech Students, Department of Computer Science and Engineering, Parul Institute of Technology, Vadodara, Gujarat, India. [2]

ABSTRACT: This paper presents a practical approach to implementing fundamental machine learning algorithms from scratch, focusing on Random Forest and K-Nearest Neighbors (KNN). The study emphasizes understanding the underlying mathematics and programming logic without relying on external libraries. The implementation is demonstrated using Python, with applications to real-world datasets. The results highlight the importance of feature engineering, data preprocessing, and model evaluation in developing efficient machine learning models. Additionally, the study discusses the advantages and challenges of manual implementation and compares the results with standard library-based implementations. This work also integrates data analysis techniques using NumPy, Pandas, Matplotlib, and Seaborn to visualize and preprocess datasets efficiently.

KEYWORDS: Machine Learning, Random Forest, K-Nearest Neighbors, Algorithm Implementation, Python, Data Analysis, NumPy, Pandas, Matplotlib, Seaborn, Feature Engineering.

I. INTRODUCTION

Machine learning algorithms are central to modern data analysis, but often, their inner workings are obscured by high-level libraries like scikit-learn. While these libraries simplify model development, implementing algorithms from scratch offers a better understanding of their mechanics. This research focuses on two popular algorithms—random forest and k-nearest neighbors (knn)—and explores their implementations without relying on pre-built functions.

By manually coding these algorithms, we gain insight into the fundamental steps involved in machine learning. The study also focuses on enhancing model performance through preprocessing techniques, such as feature scaling and handling missing values, as well as the use of advanced data analysis techniques. Furthermore, implementing these models from scratch highlights the challenges of computational complexity and optimization.

This paper is based on hands-on experience gained during an internship at intellgiere, ahmedabad, where real-world datasets were used to test the algorithms. The paper not only covers the implementation but also evaluates the effectiveness of these algorithms using metrics such as accuracy, precision, recall, and f1-score. It also highlights the importance of feature engineering and hyperparameter tuning in improving model performance.

The main objective is to show that a strong foundational understanding of machine learning algorithms can lead to better customization, debugging, and optimization of models. This approach is beneficial for anyone looking to gain deeper insight into machine learning concepts and techniques.

II. LITERATURE SURVEY

While most machine learning research and applications rely on libraries like TensorFlow and Scikit-learn, which abstract away the complexities, there is significant value in implementing algorithms from scratch. Many studies have shown that manually coding algorithms fosters a deeper understanding of their inner workings and aids in debugging and customization.

Previous research in the field has primarily focused on optimizing machine learning algorithms, often relying on standard libraries for implementation. However, these studies typically overlook the importance of understanding the

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 3, March 2025

|DOI: 10.15680/IJIRSET.2025.1403383|

underlying mathematics and logic behind the algorithms. By manually implementing Random Forest and KNN, this study offers a more educational perspective that highlights both the challenges and rewards of such an approach.
Data preprocessing and feature selection are essential components of machine learning, as they can significantly impact model performance. Several studies emphasize the role of data cleaning and the need for efficient data handling techniques. In this research, tools like NumPy and Pandas are used to preprocess the data, and Matplotlib and Seaborn are employed for visualization, helping to optimize feature selection and improve model accuracy.
Implementing algorithms from scratch also forces the exploration of more advanced optimization strategies, like hyperparameter tuning and search tree algorithms, which are often overlooked in library-based implementations. These optimizations help mitigate common issues such as overfitting and high computational complexity, which are addressed in this study.

III. METHODOLOGY

The first step in the methodology was data collection. Datasets were sourced from repositories like Kaggle and the UCI Machine Learning Repository, focusing on classification tasks. After gathering the data, it was preprocessed by handling missing values, normalizing numerical features, and splitting the data into training and testing sets. EDA was conducted using visualizations to explore feature distributions, correlations, and outliers.

The Random Forest algorithm was implemented by creating decision trees based on bootstrapped samples from the training data. Each tree was trained using information gain or Gini impurity for split decisions. The final prediction was made by aggregating the outputs of all trees using majority voting. Similarly, KNN was implemented by storing training data and using Euclidean distance to classify new data points based on their proximity to the nearest neighbors.

Data analysis was performed using Pandas for data manipulation and NumPy for handling large datasets. For visualizations, Matplotlib and Seaborn were used to plot histograms, boxplots, and pair plots, which helped to identify patterns in the data. These tools provided valuable insights into feature importance and class distribution, which in turn guided decisions regarding feature engineering and model tuning.

Evaluation metrics such as accuracy, precision, recall, and F1-score were used to assess model performance. A confusion matrix was also employed to give a detailed breakdown of classification results. The performance of the manually implemented models was compared to the results from library-based implementations to highlight the advantages and limitations of each approach.

A. Data Collection and Preprocessing

To ensure robustness in testing, datasets from publicly available repositories such as the UCI Machine Learning Repository and Kaggle were selected. These datasets included a variety of classification tasks relevant to healthcare, sentiment analysis, and customer segmentation. Preprocessing the data is crucial for removing noise, handling missing values, and ensuring that the features are correctly formatted for model training. The preprocessing steps were as follows:

1. Handling Missing Data: Missing data were addressed using imputation (mean for continuous variables, mode for categorical ones) or deletion depending on the dataset's characteristics. Pandas was used to identify and manage missing entries efficiently.
2. Feature Scaling: Since machine learning algorithms are sensitive to the scale of input features, numerical features were normalized or standardized using NumPy. This ensures that all features contribute equally to the model without one dominating due to its scale.
3. Dataset Splitting: The dataset was split into training (80%) and testing (20%) sets, ensuring that the training data was used to learn the model and the testing data was kept unseen for unbiased evaluation.
4. Exploratory Data Analysis (EDA): EDA was conducted using Matplotlib and Seaborn to understand the distribution of features, identify potential outliers, and visualize correlations between features, which informed feature selection and engineering.
5. Feature Engineering: New features were created by combining or transforming existing ones, such as calculating ratios, logarithmic transformations, or encoding categorical variables, to potentially increase model accuracy.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 3, March 2025

|DOI: 10.15680/IJIRSET.2025.1403383|

B. Algorithm Implementation

1. Random Forest

The Random Forest algorithm was implemented by first constructing multiple decision trees using bootstrapped subsets of the training data. Each decision tree was built by recursively splitting nodes based on information gain (or Gini impurity), ensuring that each tree was trained on a unique subset of the data.

Bootstrapping

Bootstrapping involves generating a new dataset by randomly sampling with replacement from the original dataset. This technique introduces diversity into the model by ensuring that different decision trees see slightly different variations of the data. Some data points may appear multiple times in a bootstrapped dataset, while others may not appear at all. This variation helps to prevent overfitting and improves the generalizability of the Random Forest model.

Decision Tree Construction

Each decision tree in the Random Forest follows a recursive process of splitting the dataset into smaller subsets based on a chosen feature and a splitting criterion. The two most commonly used criteria for this process are:

Information Gain (Entropy-Based Splitting): This method selects the feature that results in the maximum reduction of entropy, thereby increasing the purity of the child nodes.

Gini Impurity: This criterion selects the feature that minimizes the probability of misclassification by choosing the split that results in the most homogeneous child nodes.

The decision tree continues to split until a stopping criterion is met, such as a predefined depth limit, minimum number of samples per leaf, or no significant gain from further splitting.

Majority Voting for Prediction

Once multiple decision trees have been trained on their respective bootstrapped subsets, they are aggregated to make a final prediction. In the case of classification tasks, the Random Forest follows a majority voting scheme:

Each decision tree independently predicts a class label for a given input sample.

The class label that receives the most votes across all decision trees is selected as the final prediction.

This ensembling method reduces variance and increases the robustness of predictions compared to individual decision trees.

In regression tasks, the final prediction is computed by averaging the outputs from all decision trees instead of using majority voting.

2. K-Nearest Neighbors (KNN)

The K-Nearest Neighbors (KNN) algorithm classifies a data point by considering the labels of its k nearest neighbors in the training dataset. The algorithm operates in a non-parametric manner, meaning that it does not make assumptions about the underlying distribution of the data.

Distance Metric

To determine the similarity between data points, the Euclidean distance metric was employed. The Euclidean distance between two points and in an n-dimensional space is computed as: Other distance metrics, such as Manhattan distance or Minkowski distance, can also be used depending on the specific problem domain and dataset characteristics.

Classification Using Majority Voting

Once the k nearest neighbors have been identified, the algorithm assigns a class label to the test point based on the majority class among the neighbors. This process ensures that classification is driven by the local structure of the data.

For regression tasks, the prediction is calculated by averaging the target values of the k nearest neighbors instead of using majority voting.

Optimization with KD-Trees

One of the main challenges with KNN is its computational inefficiency, particularly for large datasets with high-dimensional features. To address this, the KD-Tree (K-dimensional tree) data structure is used to optimize the nearest-neighbor search. KD-Trees work as follows:

The dataset is recursively partitioned into hyperplanes based on feature values, forming a tree-like structure.

At query time, the KD-Tree is traversed to efficiently locate the nearest neighbors instead of computing distances exhaustively.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 3, March 2025

|DOI: 10.15680/IJIRSET.2025.1403383|

This significantly reduces the time complexity of the KNN algorithm from (brute force search) to (tree-based search), making it more scalable for large datasets.
By implementing these optimizations and techniques, both Random Forest and KNN can be effectively applied to a wide range of machine learning tasks, balancing accuracy and computational efficiency.

C. Data Analysis Techniques

Descriptive Statistics

Descriptive statistics were used to summarize and understand the central tendencies and variability of the features. Key statistics such as mean, median, mode, standard deviation, and interquartile range (IQR) were computed. These measures provided insights into the distribution, spread, and skewness of the data, helping identify potential outliers and inconsistencies.

Correlation Analysis

To examine relationships between features, correlation analysis was performed. A correlation matrix was computed, indicating the degree to which two variables move together. Pearson's correlation coefficient was used for continuous features, while Spearman's correlation was considered for ordinal data. To visually represent the correlation matrix, a heatmap was generated using Seaborn's heatmap function. This analysis helped in detecting multicollinearity among features, allowing for the removal of redundant variables that could negatively impact model performance.

Data Visualization

Visualization techniques were employed to explore the dataset, detect anomalies, and gain a deeper understanding of feature distributions. Several types of plots were used:

- Histograms: Displayed frequency distributions of individual features, helping to assess skewness and identify potential transformations.
- Boxplots: Used to detect outliers by showing the distribution of values through quartiles and whiskers.
- Pair Plots: Illustrated pairwise relationships between multiple features, enabling the identification of trends and clusters.
- Bar Charts: Provided a clear representation of categorical data distributions and class imbalances.
- These visualizations played a crucial role in feature selection and preparation before modeling, ensuring that the dataset was well-structured and free from inconsistencies.

D. Evaluation Metrics

To comprehensively evaluate the performance of the algorithms, the following metrics were used:

Accuracy

Accuracy measures the proportion of correctly classified instances out of the total instances. It is useful when the dataset is balanced but may not be the best metric for imbalanced datasets.

Precision and Recall

Precision and recall are particularly important for imbalanced datasets:

- Precision: Measures the accuracy of positive predictions by computing the ratio of true positive predictions to the total predicted positives.
- Recall: Measures the ability of the model to correctly identify all relevant instances by computing the ratio of true positives to the total actual positives.

F1-Score

The F1-Score is the harmonic mean of precision and recall, providing a balanced metric when dealing with class imbalance: It is particularly useful when false positives and false negatives carry different costs.

Computational Complexity

The computational cost of training and inference was measured to assess the scalability of the algorithms for larger datasets. Factors such as training time, memory usage, and prediction latency were evaluated.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 14, Issue 3, March 2025

|DOI: 10.15680/IJIRSET.2025.1403383|

Confusion Matrix
A confusion matrix was generated to provide a detailed breakdown of true positives, false positives, true negatives, and false negatives. This helped in diagnosing model weaknesses and understanding where misclassifications occurred.

IV. EXPERIMENTAL RESULTS

The results showed that the manually implemented Random Forest algorithm achieved an accuracy of 85%, while KNN achieved 80%. These results were consistent with those obtained from standard library-based models, demonstrating that the manual implementations were effective. However, KNN faced performance issues on larger datasets due to the non-optimized nearest neighbor search, which led to increased computational time.
One of the major challenges encountered during the implementation was managing the computational complexity of the algorithms. In the case of Random Forest, overfitting was addressed by pruning the decision trees and limiting their depth. For KNN, optimization techniques such as KD-Trees were applied to improve the efficiency of the nearest neighbor search and reduce latency.
Another challenge was handling missing values and outliers in the data, which can significantly affect model performance. Feature engineering played a crucial role in improving accuracy by creating new features or transforming existing ones. The integration of advanced data analysis techniques ensured that the datasets were preprocessed effectively, resulting in cleaner input for the machine learning models.
Despite these challenges, the manually coded models performed well, and the study demonstrates the value of implementing algorithms from scratch to gain a deeper understanding of machine learning. It also provides a comparison with library-based implementations, showing that manual coding can be an educational and effective approach for building machine learning models.

Model	Accuracy	Precision	Recall	F1-Score
Random Forest	85%	0.82	0.84	0.83
K-Nearest Neighbors	80%	0.78	0.76	0.77

V. CONCLUSION

This study demonstrates that implementing machine learning algorithms like Random Forest and KNN from scratch provides valuable insights into their inner workings. By focusing on the foundational principles, this paper offers an educational perspective that highlights the importance of data preprocessing, feature engineering, and model evaluation. The results indicate that manual implementations can achieve performance similar to library-based models, although they may be less efficient in handling large datasets.
Key takeaways from this study include the importance of hyperparameter tuning, feature selection, and optimization techniques for improving model performance. Future work will involve implementing additional models such as Support Vector Machines (SVM) and Neural Networks to further expand the understanding of machine learning algorithms. Further optimization of the existing models will also be explored to handle large datasets more efficiently.
There is also potential for applying these algorithms to more complex, real-world datasets in fields like finance, healthcare, and e-commerce. Additionally, future work will involve exploring deep learning techniques for tasks that require higher computational power and more complex model structures.
Overall, this paper highlights the value of hands-on experience in understanding and implementing machine learning algorithms and sets the foundation for further research in the area.

REFERENCES

[1] L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5-32, 2001.
[2] T. Cover and P. Hart, "Nearest Neighbor Pattern Classification," IEEE Transactions on Information Theory, vol. 13, no. 1, pp. 21-27, 1967.
[3] I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning," MIT Press, 2016.
For code and datasets, visit: https://github.com/Purvang2803

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi cross ref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details