Date of publication August 27, 2023

Digital Object Identifier 10.1109/ACCESS.2017.DOI

K Nearest Neighbour(KNN) classification on Statlog (Landsat Satellite) Dataset

RASHMI KHADKA[1] (THA076BCT035), SHIWANI SHAH[2] (THA076BCT042)
[1]Department of Electronics and Computer Engineering, Thapathali Campus (e-mail: rashmikhadka6255@gmail.com)
[2]Department of Electronics and Computer Engineering, Thapathali Campus (e-mail: shahshiwani70@gmail.com)

ABSTRACT This study investigates the effectiveness of two variants of the K-Nearest Neighbors (KNN) classifier, the default KNN, and distance-weighted KNN, using the "Statlog (Landsat Satellite) Dataset" for land cover classification based on satellite imagery. The dataset consists of multi-spectral values of pixels in 3x3 neighborhoods within satellite images, with each pixel associated with a specific land cover class. The objective is to compare the performance of these KNN variants in accurately classifying land cover types. The default KNN employs a straightforward majority vote scheme, while distance-weighted KNN assigns weights to neighboring pixels based on their distances. Through data preprocessing, model training, and evaluation using classification metrics, we assess and contrast the predictive capabilities of both KNN variants. This analysis provides insights into the suitability of each KNN approach for satellite-based land cover classification, aiding in better decision-making for remote sensing applications.

INDEX TERMS Default KNN, Distance-Weighted KNN,Euclidean distance,K Nearest Neighbour, KNN, Manhatthan distance, Normalization, Statlog (Landsat Satellite) Dataset

I. INTRODUCTION

LAND cover classification plays a pivotal role in remote sensing and environmental studies, facilitating critical tasks such as monitoring agricultural development, urban planning, and natural resource management. In this context, accurate classification of satellite imagery is paramount. The K-Nearest Neighbors (KNN) algorithm, known for its simplicity and efficacy, offers a promising approach to this challenging problem. This study embarks on an exploration of two variants of the KNN classifier: the default KNN and distance-weighted KNN, applied to the "Statlog (Landsat Satellite) Dataset."

The "Statlog (Landsat Satellite) Dataset" presents a rich repository of multi-spectral values of pixels in 3x3 neighborhoods within satellite images, each associated with a specific land cover class. While the default KNN relies on a straightforward majority vote scheme among nearest neighbors, distance-weighted KNN introduces an innovative weighting mechanism based on the proximity of neighboring pixels. This study seeks to elucidate the comparative performance of these KNN variants in accurately classifying land cover types within satellite imagery.

Its objective is to inform the best practices in remote sensing applications, aiding in more precise land cover mapping and bolstering decision-making in diverse domains. and to use knowledge surrounding KNN classifiers' suitability for land cover classification, particularly in the context of satellite imagery analysis. By comprehensively evaluating the default KNN and distance-weighted KNN, we aspire to provide valuable insights for researchers and practitioners in remote sensing, environmental monitoring, and land management.

II. METHODOLOGY

A. K NEAREST NEIGHBOUR ALGORITHM

The k-Nearest Neighbors (KNN) algorithm is a decision-making system, quite similar to how people make decisions in everyday life. It can be considered similar as a voting system, where the majority class label determines the class label of a new data point among its nearest 'k' (where k is an integer) neighbors in the feature space. Now, let's say you have a group of neighbors, say five of them, and most of them support a particular party, let's call it Party A. In this scenario, it's quite likely that you'd also choose to vote for Party A because it's the choice of most of your nearest neighbors.

This concept is analogous to how the kNN algorithm functions. It deals with classifying data points, like different types of fruits.

Imagine you have data about two types of fruits, grapes, and pears, and you measure two features of these fruits: how round they are and their diameter. You can plot these measurements on a graph. Now, if someone hands you a new fruit, you can also plot it on the same graph. To determine what type of fruit it is, you measure the distance from this new fruit to the nearest 'k' (where 'k' is just a number) data points on the graph. If we choose 'k' to be three, we look at the three closest points to our new fruit. Let's say all three of these nearest points are pears. In this case, we can be quite confident, 100 percent, that the new fruit is a pear. However, if we set 'k' to four, we'd look at the four nearest points. If three of them are pears and one is a grape, we might say we're 75 percent sure it's a pear because the majority (three out of four) still suggests it's a pear.

- **Selection of K**
  Too small k might be inaccurate and sensitive to noise Larger k may improve performance but too large k destroys locality, i.e end up looking at samples that are not neighbors. And lose sensitivity to changes in the feature space.

B. DISTANCE MATRICS

The distance between neighbors in KNN refers to the distances between a data point of interest (the one you want to classify) and its 'k' nearest neighbors within the dataset. The goal is to identify the 'k' data points that are most similar (closest) to the point you want to classify. Two common distance metrics used in KNN are:

- **Euclidean Distance**
  This is the most widely used distance metric in KNN. It calculates the straight-line distance between two data points in the feature space. For two points, A and B, with coordinates (x1, y1) and (x2, y2), respectively, the Euclidean distance is given by:

$$\text{Euclidean Distance} = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{1}$$

  Euclidean distance treats each features as equally important.
- **Manhatthan Distance** Also known as the city block distance, this metric calculates the distance by summing the absolute differences between the coordinates of two points along each dimension. For points A and B, it is computed as:

$$\text{Manhattan Distance} = \sum_{i=1}^{n}|x_i - y_i| \tag{2}$$

C. NORMALIZATION OF VARIABLES

Normalization is a data preprocessing techniques used to adjust the range or distribution of feature values in a dataset. It insures that all features in the dataset have a similar scale and distribution, preventing certain features from dominating the distance calculations, which can be crucial in the KNN algorithm. Scaling and normalization also help improve the convergence of optimization algorithms and make the model more robust.

Normalization scales the features to a specific range, typically [0, 1]. The formula for min-max scaling is as follows:

$$X_{\text{scaled}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}} \tag{3}$$

D. DEFAULT KNN

It is the traditional KNN algorithm through Euclidean distance and majority classification rules. Specifically, it first finds the K nearest neighbors of the test data from the training dataset according to the Euclidean distance measurement. Then it takes the class with the highest frequency among the K neighbors as the class label of the test data.

To apply Default KNN for classification, follow these steps:

1) Select the number K of neighbors.
2) Calculate the Euclidean distance (or chosen distance metric) for all data points.
3) Choose the K nearest neighbors based on calculated distances.
4) Count data points in each category among the K neighbors.
5) Assign the new data point to the category with the most neighbors.
6) The default KNN model is ready.

E. DISTANCE-WEIGHTED KNN

This method is an improved KNN algorithm, which applies a weighted classification rule. After obtaining the K neighbors of the test data, it sets the weight of each neighbor according to the distance. The closer the neighbor has the greater the weight, it has the greatest impact on the label of the test data. i.e., this method is different from traditional KNN (the voting weight of each neighbor is the same), it gives different voting weights to different neighbors.

To apply weighted kNN model for classification, follow these steps:

1) Select the number K of neighbors.
2) Calculate the Euclidean distance or Manhatthan distance (or chosen distance metric) for all data points.
3) Choose the K nearest neighbors based on calculated distances.
4) Assign weights to each neighbor based on distances.
5) Calculate weighted counts of data points in each category among the K neighbors.
6) Assign the new data point to the category with the highest weighted count.
7) The distance-weighted KNN model is ready.

F. STEPS TO IMPLEMENT THE K-NEAREST NEIGHBOUR (KNN) ALGORITHM

- Data Pre-processing step

1. Handle Missing Values: Check for any missing values in the dataset and apply imputation or removal of instances with missing values.
2. Normalize Numerical Features: If the features are of different scales, normalize them to bring them to a common scale.
3. Encode Categorical Class Labels: If the class labels are categorical, encode them into numerical values to be used in the Naive Bayes classifier.

- Fitting the K-NN algorithm to the Training set

1. Split the dataset into a training set and a testing set. The training set is used to train the KNN model
2. Select an appropriate value for the number of neighbors, K, which determines how many nearby data points are considered when making predictions
3. Apply the KNN Algorithm: Implement the KNN algorithm using the training set. The model will learn from this data and store it for future predictions.

- Predicting the test result

1. Use the trained KNN model to predict the class
2. labels for the instances in the testing set.
3. Calculate the distances from each testing instance to the data points in the training set.
4. Select the K nearest neighbors for each testing instance based on the calculated distances.
5. Assign class labels to the testing instances based on a majority vote among their K nearest neighbors.

- Test Accuracy of the Result (Creation of Confusion Matrix)

1. Compare the predicted class labels with the actual class labels in the testing set.
2. Create a confusion matrix to visualize the performance of the classifier. The confusion matrix compares the predicted class labels against the actual class labels and shows the number of true positives, false positives, true negatives, and false negatives.
3. Make classification report.

- Visualizing the Test Set Result

1. For visualizations, consider 2D or 3D plots if the data has multiple features.
2. Plot the data points in the testing set along with the decision boundary determined by the KNN classifier. This visualization helps understand how well the classifier separates different classes based on the feature.

The metrics below are commonly used to evaluate the performance of classification models, including KNN classification. They provide insights into the model's accuracy, precision, recall, and overall predictive capability.

Confusion Matrix:

	Predicted Class	
Actual Class	Positive	Negative
Positive	TP	FN
Negative	FP	TN

Accuracy:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \tag{4}$$

Precision:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{5}$$

Recall:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{6}$$

F1 Score:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{7}$$

G. SYSTEM ARCHITECTURE

The system architecture for the Statlog (Landsat Satellite) Dataset is shown in the FIGURE 1

The system architecture for implementing KNN classification models on the Statlog (Landsat Satellite) dataset involves a series of well-defined steps. These steps encompass data preprocessing, feature selection, model training, prediction, and the analysis of results. Each stage contributes to the overall process of accurately classifying land cover types based on satellite imagery data.

- **Dataset**:
  The Statlog (Landsat Satellite) Data Set forms a fundamental component within our system architecture. This dataset is pivotal for its role in training and testing machine learning models. It encompasses a range of features that offer valuable insights into the characteristics of satellite imagery. These features include attributes like spectral values, which represent different bands of the Landsat satellite data, pixel neighborhoods, and classification labels for land cover types.
- **Data Preprocessing**:
  Data preprocessing ensures the dataset is cleaned, transformed, and prepared for model training. This involves handling missing values, encoding categorical features, and scaling/normalizing numerical features. Preprocessing ensures that the data is in a suitable format for the subsequent stages. For our dataset there were no missing value so no cleaning is needed.
- **Training Data and Testing Data**:
  The dataset is split into training and testing sets. The training data is used to train the Naive Bayes classifier, while the testing data is used to evaluate its performance. The typical split ratio is 80
- **Fitting KNN algorithm**:
  Two KNN models are trained: the default KNN and the distance-weighted KNN.

For the default KNN, Euclidean distance is calculated between instances.When predicting the class label for a data point, the default KNN considers the distances between that point and its 'k' nearest neighbors, where 'k' is a user-defined parameter. The majority class among these nearest neighbors determines the predicted class for the data point.
For the distance-weighted KNN model, we introduce a custom distance metric that assigns weights to neighboring data points based on their distances. This means that not all neighbors have equal influence on the prediction. Closer neighbors may have higher weights, and farther neighbors lower weights. The custom distance metric will be defined to implement this distance weighting.For this model, Manhattan distance is used.
There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.

- **Prediction**:
  Once the KNN models are trained, they are employed to predict the land cover types of the testing data instances. The models calculate distances between instances and their nearest neighbors in the training data. The class labels of the K nearest neighbors are considered, and the majority class determines the prediction.
- **Analysis of Results**:
  The final step involves analyzing the predictive performance of the KNN models. Metrics such as accuracy, precision, recall, and the F1-score are computed to evaluate the models' ability to correctly classify land cover types. The confusion matrix visually represents true positive, true negative, false positive, and false negative predictions.
- **Optimal K Value Determination**:
  To fine-tune the KNN models, an optimal K value is determined. The models are evaluated across different K values, and an accuracy vs. K value graph is plotted. The K value that maximizes accuracy is chosen for the final models.

H. INSTRUMENTATION

The implementation of Naive Bayes Classifier utilized several Python libraries to perform data analysis, model training, and evaluation. Here's a brief explanation of each library:

- **Pandas**: Pandas is a powerful data manipulation library that was used to handle the dataset. It allowed us to read, clean, and preprocess the data efficiently using DataFrames, making data exploration and transformation easier.
- **NumPy**: NumPy is a fundamental library for numerical computing in Python. It provided support for mathematical operations on arrays and matrices. It was used implicitly by Pandas for efficient data storage and processing.
- **Matplotlib**: It is a widely used visualization library in Python. It helped us create various plots, such as histograms, bar charts, and scatter plots, to visualize the distribution of data and gain insights from the dataset.
- **Seaborn**: Seaborn is a higher-level data visualization library that complements Matplotlib. It allowed us to create visually appealing statistical graphics and heatmaps. We used Seaborn to visualize the correlation matrix and feature distributions.
- **Scikit-learn**: Scikit-learn is a popular machine learning library in Python. It provided essential tools for data preprocessing, model training, and evaluation. We utilized LabelEncoder to convert categorical features into numerical labels and KBinsDiscretizer for discretizing continuous features.
- **KNeighborsClassifier:**: The KNeighborsClassifier class from Scikit-learn was used to implement the KNN algorithm. It enabled the training of KNN models on the dataset, including specifying the number of neighbors (K).
- **PCA (Principal Component Analysis)**: PCA, available in Scikit-learn, was employed for dimensionality reduction when visualizing the dataset with multiple features. PCA reduced the feature dimensions for effective visualization.
- **Confusion Matrix and Classification Report**: The confusion matrix and classification report metrics from Scikit-learn were instrumental in evaluating the KNN model's performance. The confusion matrix provided insights into true positives, true negatives, false positives, and false negatives, while the classification report offered precision, recall, F1-score, and support for each class in the target variable.

III. RESULT

A. DATASET

This is the summary of Statlog (Landsat Satellite) dataset:

Data Type	Multivariable
Number of Attributes	36
Number of instances	6,435
Attribute Type	Integer
Missing Values	No
Subject Area	Physical Science

TABLE 1: Dataset Summary

The classes are represented as numerical codes:

1. Red Soil
2. Cotton Crop
3. Grey Soil
4. Damp Grey Soil
5. Soil with Vegetation Stubble
6. Mixture Class (All Types Present)
7. Very Damp Grey Soil

(Note: There are no examples with class 6 in this dataset) Each instance in the dataset represents a 3x3 square neighborhood of pixels within a sub-area of a Landsat satellite image. The dataset contains 36 attributes in total. Each attribute represents a spectral value, ranging from 0 to 255, for different pixels within the 3x3 neighborhood. The four spectral values for the central pixel are given by attributes Q, R, S and T.

B. ANALYSIS OF KNN CLASSIFICATION ALGORITHM IN STATLOG (LANDSAT SATELLITE) DATASET

The comprehensive analysis of the performance and effectiveness of the KNN classification algorithm in Statlog (Landsat Satellite) dataset is presented here. We aim to understand how KNN works, its performance metrics, and the impact of different parameters such as the choice of distance metric and the value of "k." Figure 2 visualizes the label distribution within the Statlog (Landsat Satellite) dataset. Figure 3 offers a 2D visualization, providing insights into the dataset's structure.

The preprocessing steps were undertaken to prepare the data for model training. This includes handling missing values, encoding categorical features, and scaling numerical features. Next, we delve into the model training phase, where we employ both default KNN and Distance-weighted KNN. Subsequently, we present the predictions made by the trained models on the testing data. We discuss the predicted soil type of the first 5 samples and compare them with their corresponding ground truth labels.

To gain deeper insights into the performance of the KNN models, we analyze their respective confusion matrices. This analysis allows us to examine the true positive, true negative, false positive, and false negative predictions, offering a comprehensive view of the classifier's precision, recall, and F1-score.

Figure 4 displays the confusion matrix for the Default KNN model, allowing us to assess its performance. Figure 5 presents the distribution of features with continuous values in the Default KNN model. Figure 6 provides a further look into the distribution of features with continuous values in the Default KNN model. Figure 7 presents the classification report for the Default KNN model, offering detailed performance metrics.

Figure 8 displays the confusion matrix for the Distance-Weighted KNN model.

Figure 9 presents the correlation matrix of the Distance-Weighted KNN model.

Figure 10 offers the classification report for the Distance-Weighted KNN model.

We also provide a comprehensive analysis of the K-nearest neighbors (KNN) classification algorithm applied to the Statlog (Landsat Satellite) dataset.

Figure 13 depicts the graph illustrating the relationship between accuracy and the value of "k" in the KNN model.

By this analysis the best or optimum value of K was found to be 5 for our dataset. Using this K value an Optimum KNN model is trained and analysis same as before.

Figure 11 displays the confusion matrix for the Optimum KNN model.

Figure 12 presents the classification report for the Optimum KNN model.

Each of these figures contributes to our comprehensive analysis of the KNN classification algorithm's performance on the Statlog (Landsat Satellite) dataset.

IV. DISCUSSION AND ANALYSIS

In this study, we applied two variants of the K-Nearest Neighbors (KNN) classification algorithm: the default KNN and distance-weighted KNN, to address a classification problem using the Statlog (Landsat Satellite) dataset. Both models were employed to classify multi-spectral values of pixels in 3x3 neighborhoods in satellite images, aiming to predict the central pixel's classification. While KNN is known for its simplicity and versatility, these two variants had distinct characteristics and performance.

The default KNN classifier assigns class labels based on a majority vote among the K nearest neighbors. It assumes that all neighbors contribute equally to the decision, regardless of their distances. This model performed reasonably well, achieving a certain level of accuracy in predicting the central pixel's class. However, it might have limitations when dealing with datasets where the influence of closer neighbors should be prioritized.

On the other hand, the distance-weighted KNN classifier introduced a more nuanced approach. Instead of treating all neighbors equally, it assigned weights to each neighbor based on their distances. Closer neighbors had a stronger influence on the prediction, which could be advantageous in cases where nearby data points carry more relevant information. This model showed promise in enhancing the classification accuracy, especially when the dataset exhibited varying distances among neighbors.

The choice between the default KNN and distance-weighted KNN depends on the dataset's characteristics and the problem at hand. For datasets where proximity plays a critical role in classification, the distance-weighted KNN may offer better results. However, it's important to consider the computational complexity introduced by assigning and calculating weights for each neighbor. In the context of the Statlog (Landsat Satellite) dataset, the distance-weighted KNN model demonstrated improved performance, showcasing its potential for applications involving satellite image classification.

V. CONCLUSION

In conclusion, both the default KNN and distance-weighted KNN classifiers were explored in the context of classifying multi-spectral values in satellite imagery using the Statlog (Landsat Satellite) dataset. While the default KNN provides a straightforward approach, the distance-weighted KNN introduced a valuable variation that accounts for the influence of distances among neighbors. For the Statlog (Landsat Satellite) dataset, the distance-weighted KNN model displayed enhanced accuracy in predicting the central pixel's classification compared to the default KNN. This suggests that considering the distances between data points can significantly impact the model's performance, particularly in scenarios where spatial proximity is essential.

This study provides insights into the potential of KNN-based approaches for satellite image classification and offers guidance for optimizing classification accuracy in remote sensing applications. Further research can explore additional distance metrics and fine-tuning of hyperparameters to further enhance the performance of KNN models in satellite image analysis.

References

[1] D. Lewis, *"KNN Classification with One-step Computation"*, Shichao Zhang, Senior Member, IEEE, Jiaye Li*.
[2] *Statlog(landsat Satellite) Dataset*, [Online]. Available: https://archive.ics.uci.edu/dataset/146/statlog+landsat+satellite.
[3] *scikit-learn* K-NeighborsClassifier. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html.
[4] *K-Nearest Neighbors Algorithm*, [Online]. Available: https://www.ibm.com/topics/knn.

RASHMI KHADKA is a driven individual currently pursuing a Bachelor's degree in Computer Engineering at Thapathali Campus. She possesses a strong passion for machine learning and data mining, consistently seeking to delve into the latest advancements in these domains. While Rashmi may not have amassed notable achievements at this point, her enthusiasm and determination to learn and apply state-of-the-art technologies make her a promising and ambitious figure in the field of computer engineering.

SHIWANI SHAH is a dedicated individual currently studying Bachelor's in Computer Engineering at Thapathali Campus. With a strong passion for research and innovation, Shiwani actively engages in various projects and practical applications to apply her knowledge. Her continuous quest for learning drives her to stay updated with the latest advancements and trends in the field. Equipped with expertise in machine learning and data science, Shiwani's relentless determination, inquisitive mindset, and commitment to technology position her to make significant contributions to the ever-evolving landscape of the industry.

• • •

APPENDIX A: FIGURES

A. FIGURES

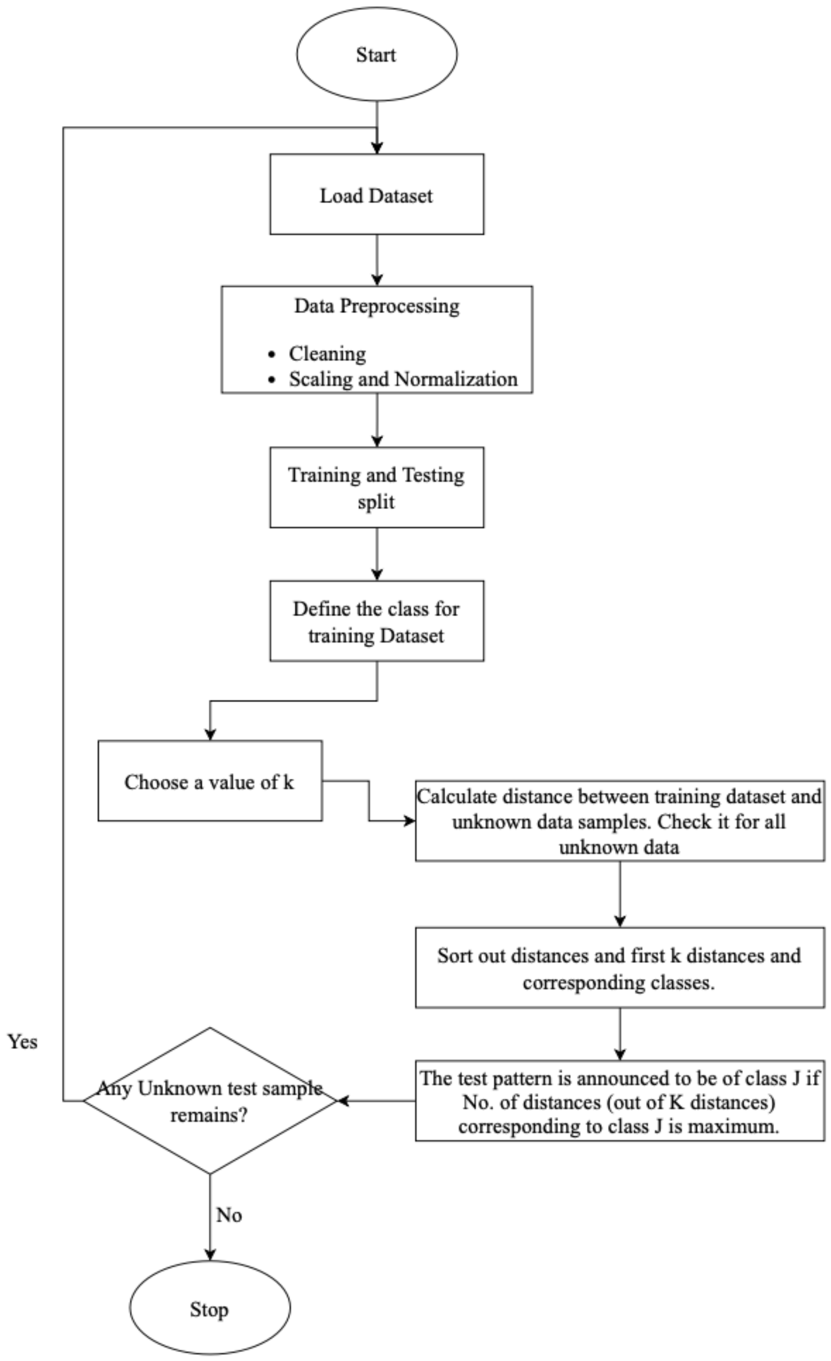

Start
Load Dataset
Data Preprocessing
• Cleaning
• Scaling and Normalization
Training and Testing split
Define the class for training Dataset
Choose a value of k
Calculate distance between training dataset and unknown data samples. Check it for all unknown data
Sort out distances and first k distances and corresponding classes.
The test pattern is announced to be of class J if No. of distances (out of K distances) corresponding to class J is maximum.
Any Unknown test sample remains?
Yes
No
Stop

FIGURE 1: Flowchart for KNN algorithm

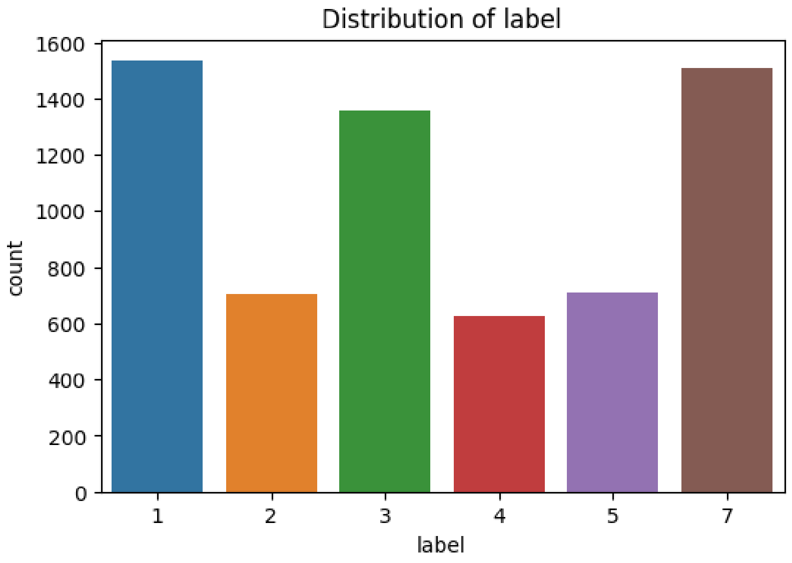


FIGURE 2: Label distribution visualization of Statlog(landsat satellite) Dataset

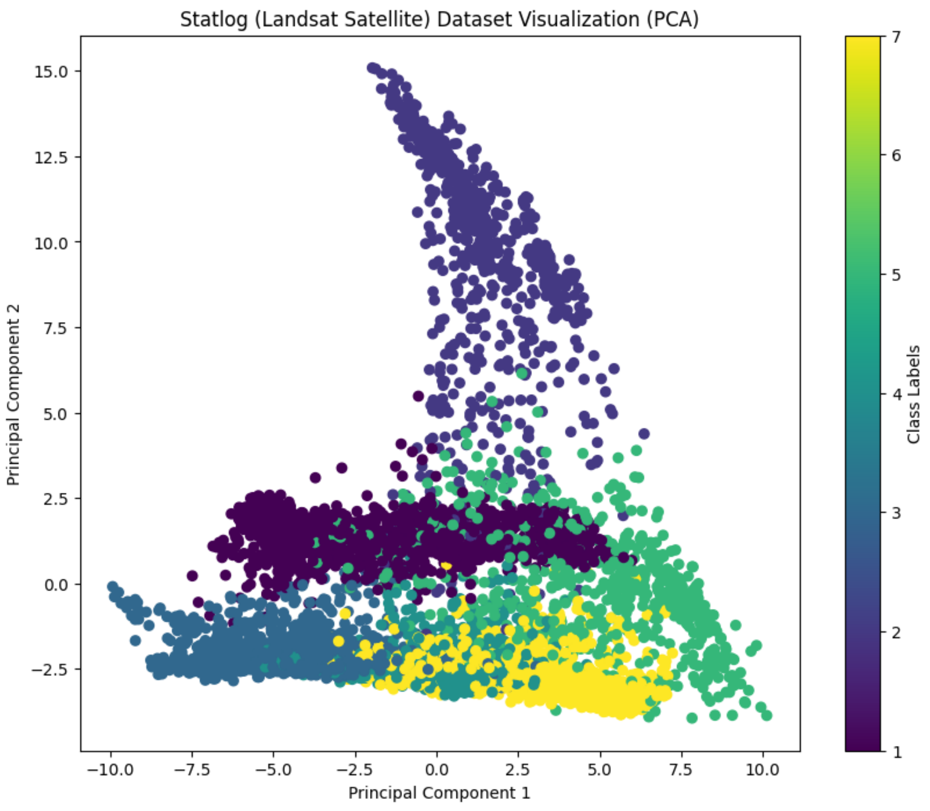


FIGURE 3: 2D Visualization

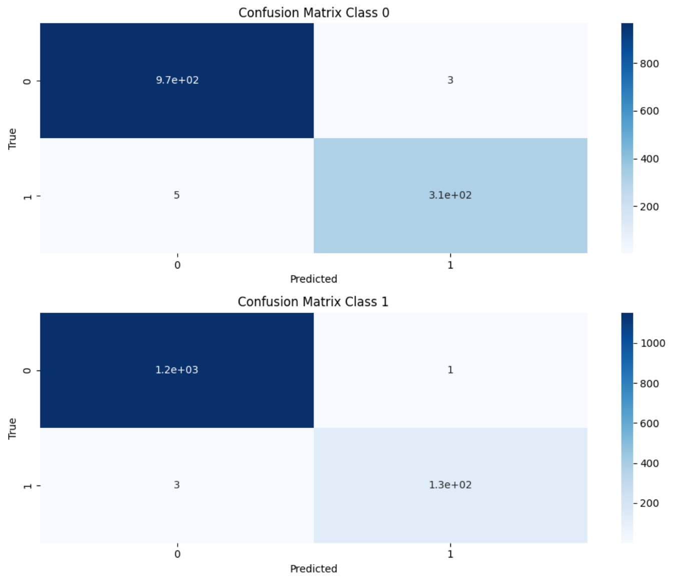


FIGURE 4: Confusion matrix of Default KNN

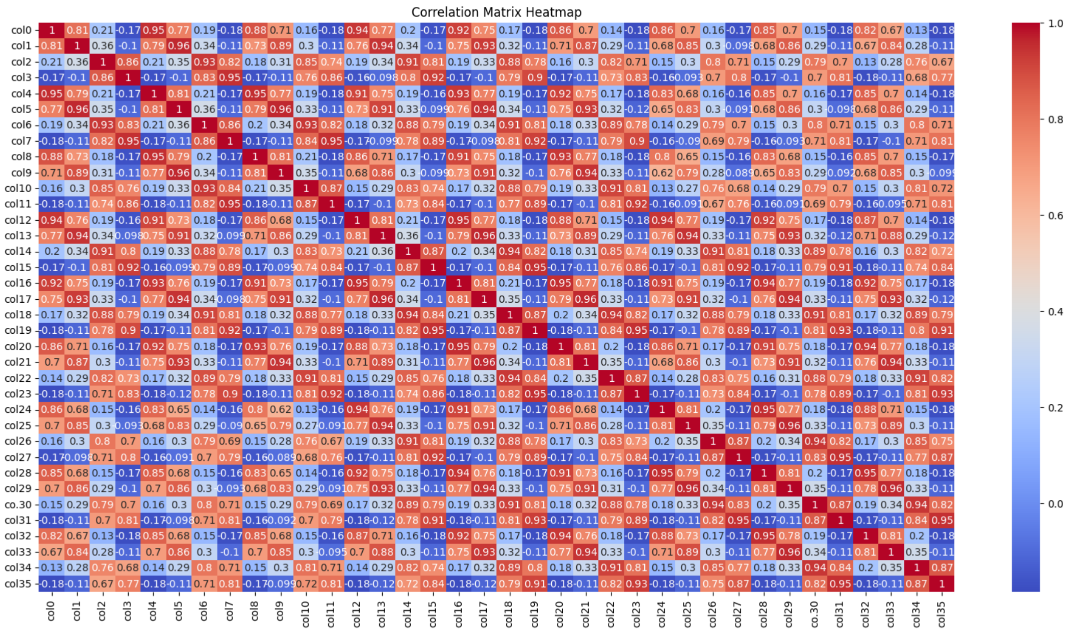


FIGURE 5: Distribution of features with continuous values

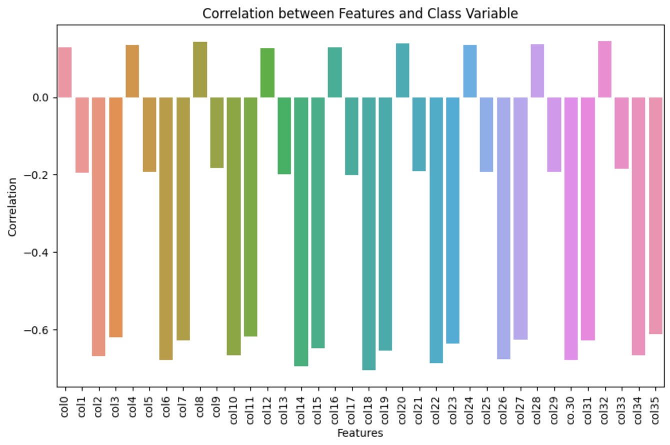
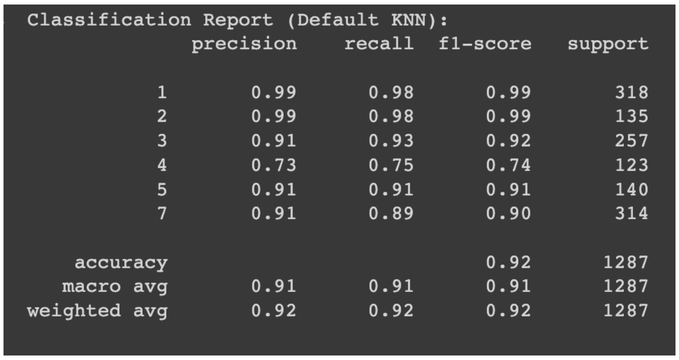


FIGURE 6: Distribution of features with continuous values



```
Classification Report (Default KNN):
              precision    recall  f1-score   support

           1       0.99      0.98      0.99       318
           2       0.99      0.98      0.99       135
           3       0.91      0.93      0.92       257
           4       0.73      0.75      0.74       123
           5       0.91      0.91      0.91       140
           7       0.91      0.89      0.90       314

    accuracy                           0.92      1287
   macro avg       0.91      0.91      0.91      1287
weighted avg       0.92      0.92      0.92      1287
```

FIGURE 7: Classification Report of Default KNN

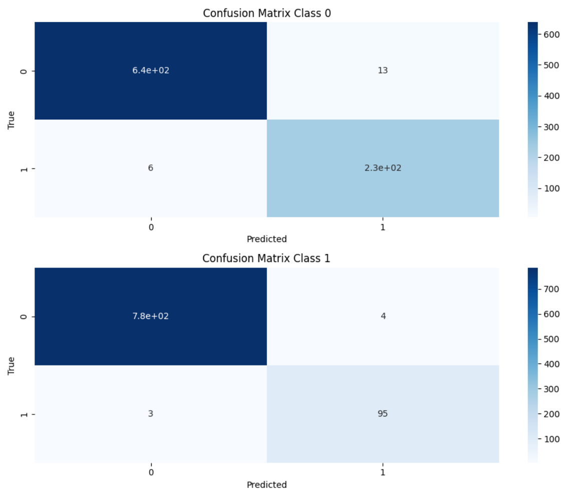


FIGURE 8: Confusion matrix for Distance-Weighted KNN

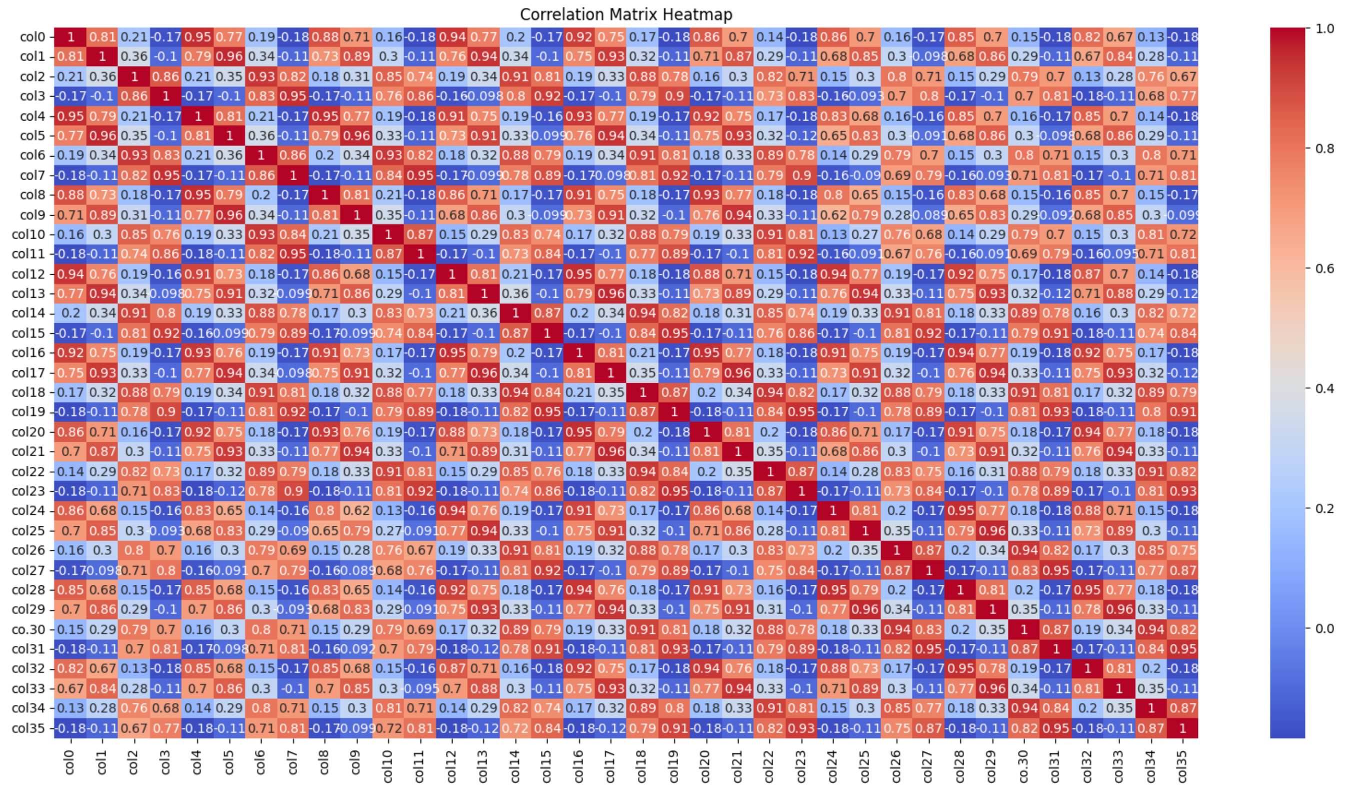
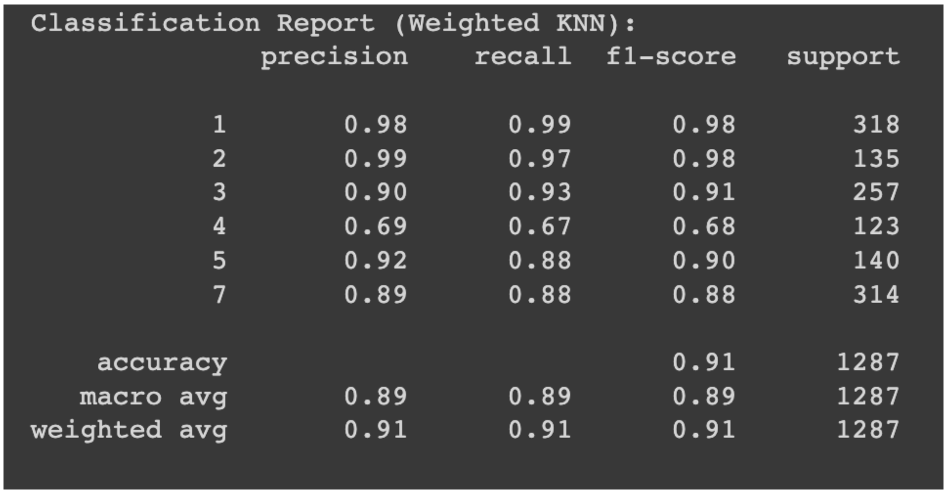


FIGURE 9: Correlation matrix of Distance-Weighted KNN



```
Classification Report (Weighted KNN):
             precision    recall  f1-score   support

           1       0.98      0.99      0.98       318
           2       0.99      0.97      0.98       135
           3       0.90      0.93      0.91       257
           4       0.69      0.67      0.68       123
           5       0.92      0.88      0.90       140
           7       0.89      0.88      0.88       314

    accuracy                           0.91      1287
   macro avg       0.89      0.89      0.89      1287
weighted avg       0.91      0.91      0.91      1287
```

FIGURE 10: Classification Report of Distance-Weighted KNN

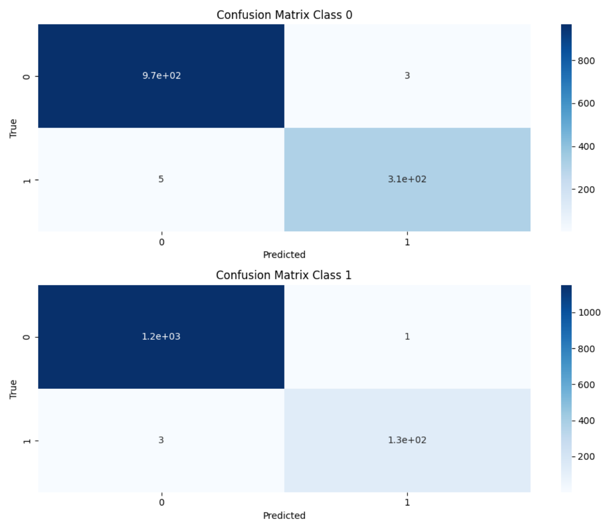
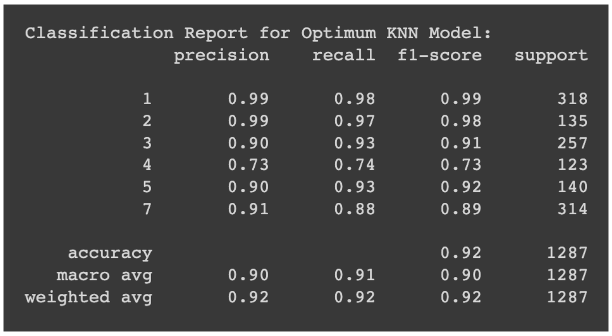


FIGURE 11: Confusion matrix for Optimum KNN



```
Classification Report for Optimum KNN Model:
              precision    recall  f1-score   support

           1       0.99      0.98      0.99       318
           2       0.99      0.97      0.98       135
           3       0.90      0.93      0.91       257
           4       0.73      0.74      0.73       123
           5       0.90      0.93      0.92       140
           7       0.91      0.88      0.89       314

    accuracy                           0.92      1287
   macro avg       0.90      0.91      0.90      1287
weighted avg       0.92      0.92      0.92      1287
```

FIGURE 12: Classification report for Optimum KNN

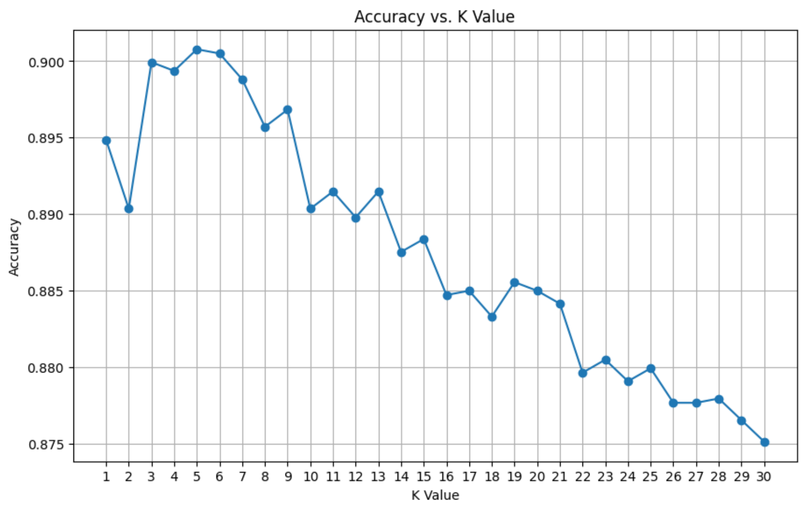


FIGURE 13: Accuracy VS K value graph