KNN Implementation from Scratch

Explanation of the KNN Algorithm and Its Implementation

The **K-Nearest Neighbors (KNN)** is an algorithsm that predicts the class of a feature based on the class of other instances with the nearest input features.

Steps in KNN Implementation:

1. **Distance Calculation**: Given a query point, the algorithm computes the distance (in this case using Euclidean distance) between this point and all training data points.
2. **Neighbor Identification**: The algorithm sorts the calculated distances from the query point to all other training points to identify the k-nearest neighbors.
3. **Majority Voting**: The predicted class for the test point is determined by majority voting amongst the k-nearest neighbors.

Description of the Optimal k Value and Its Justification

The implemented KNN code evaluates various values of k ranging from 1 to 20 to find the most optimal k-value based on accuracy, for that instance of the code. Based on running the code manually, the values in the range of 3-5 seem to come up most often which makes sense given the dataset.

Justification:

- A small k-value can lead to overfitting, as it becomes over sensitive to noise in the data, while a large k-value may underfit and overly smooth out class boundaries.
- The optimal range of the k-value being 3-5, provides a good balance between over and underfitting, allowing the model to better generalize and accurately classify tumor samples as benign or malignant.

Evaluation Metrics and Their Interpretation

The performance of KNN in this first question was assessed using accuracy, which measures the proportion of correctly classified instances against the total instances. Higher accuracy indicates better model performance in the prediction of benign versus malignant tumors.

In this evaluation, KNN achieved very high accuracy scores consistently over 95% with regard to the optimal k-value, reflecting its effectiveness in selecting the right class between malignant and benign tumors in the dataset.

Logistic Regression using sklearn

Explanation of Logistic Regression and Its Implementation Using sklearn

Logistic Regression is a statistical method used for tasks where the outcome can be true or false, in this case malignant or benign tumors. Logistic regression predicts the probability that a given input point belongs to a particular class by applying the logistic function.

Steps in Logistic Regression Implementation:

1. **Model Initialization**: A logistic regression model is created using the LogisticRegression class from the sklearn library.
2. **Fitting the Model**: The model is fitted on the training dataset using the fit() method, which adjusts the model parameters to best fit the data.
3. **Making Predictions**: Predictions on the test dataset are made using the predict() method, which outputs the class labels based on the predicted probabilities.

Evaluation Metrics and Their Interpretation

The performance of the Logistic Regression model was evaluated using the following key metrics:

Accuracy: Represents the proportion of correctly classified instances against the total instances. A higher accuracy indicates better model performance when predicting benign versus malignant tumors.

Confusion Matrix: A table summarizing the prediction results. It displays true positives, true negatives, false positives, and false negatives, which helps understand the model's performance when it comes to accurately determining the class of the tumors.

Precision: The proportion of true positive predictions versus all positive predictions. High precision indicates a low rate of false positives.

Recall: Measures the proportion of true positives among all actual positives in the dataset. High recall reflects the ability to capture all relevant instances.

F1 Score: The mean of precision and recall, providing a single metric that balances both concerns.

Comparison of Logistic Regression and KNN Performance

The performances of Logistic Regression and KNN were compared based on:

- **Accuracy**: Both models had similarly high accuracy, but Logistic Regression very slightly outperformed KNN at the optimal k-value.
- **Precision and Recall**: Logistic Regression showed slightly higher precision, indicating better performance in avoiding false positives, while KNN had marginally better recall, suggesting effectiveness in capturing all positive instances.
- **F1 Score**: The F1 score for KNN was ever so slightly better than that of Logistic Regression, suggesting it strikes a better balance between precision and recall.