**K-Nearest Neighbors (KNN) Implementation for Abalone Age Prediction**

**Introduction**

This report discusses the implementation of the K-Nearest Neighbors (KNN) algorithm for predicting the age of abalone based on physical measurements. The project was completed as part of Assignment 1 for the Machine Learning course.

**Code Overview**

The Python script implements a custom KNN algorithm and applies it to the Abalone dataset. The main components of the code include:

1. Data preprocessing

2. Custom KNN class implementation

3. Model evaluation and metrics calculation

4. Hyperparameter tuning (finding the optimal K value)

5. Visualization of results

**Data Preprocessing**

The Abalone dataset is loaded using pandas, with the following steps:

1. The 'Sex' column is dropped as it's considered non-impactful for age prediction.

2. The dataset is split into features (X) and target variable (y).

3. The data is further split into training and test sets using a 70-30 ratio.

**Custom KNN Implementation**

**A custom KNN class is implemented with the following methods:**

- \_\_init\_\_: Initializes the KNN model with a specified K value.

- fit: Stores the training data.

- predict: Predicts the target variable for multiple input samples.

- \_predict: Helper method to predict for a single sample using Euclidean distance.

**Model Evaluation**

The script includes functions to calculate various performance metrics:

- Accuracy

- Precision

- Recall

- F1-score

These metrics are calculated for both training and test sets to evaluate the model's performance.

**Hyperparameter Tuning**

The script implements a function find\_best\_k to determine the optimal K value:

1. It iterates through a range of K values (1 to 99, odd numbers only).

2. For each K, it trains the model and calculates performance metrics.

3. The K value with the highest test accuracy is selected as the best K.

**Visualization**

The relationship between K values and accuracies (both training and test) is visualized using matplotlib. This helps in understanding the model's performance across different K values.

**Challenges and Improvements**

1. Computational Efficiency: The custom KNN implementation may be slower than optimized libraries for large datasets. Future improvements could include using more efficient data structures or algorithms for nearest neighbor search.

2. Feature Scaling: The current implementation doesn't include feature scaling, which could be important given the different scales of the physical measurements. Adding normalization or standardization could potentially improve the model's performance.

3. Handling Ties: The current implementation doesn't explicitly handle ties in voting. Implementing a tie-breaking mechanism could make the model more robust.

4. Cross-Validation: The script uses a single train-test split. Implementing k-fold cross-validation could provide a more reliable estimate of the model's performance.

5. Feature Importance: The current implementation treats all features equally. Analyzing feature importance and potentially implementing feature selection or weighting could improve the model's accuracy.

**Results and Conclusion**

**The script outputs the following:**

1. Performance metrics for each K value tested

2. The best K value found

3. Detailed metrics for the model with the best K value on both training and test sets

4. A plot showing the relationship between K values and accuracies

These results allow for a comprehensive analysis of the KNN model's performance on the Abalone dataset. The visualization helps in understanding the trade-off between model complexity (K value) and performance.

In conclusion, this implementation provides a solid foundation for understanding and applying the KNN algorithm to real-world datasets. The custom implementation, coupled with thorough evaluation and visualization, offers valuable insights into the workings of the KNN algorithm and its application to the problem of abalone age prediction.