PROJECT PRESENTATION

ENHANCED KNN USING BAGGING AND WEIGHTS

Alexandre Teixeira, 202206427
Guilherme Oliveira, 202204987
Mário Minhava, 202206190

EXECUTIVE SUMMARY

1. Goals

- Primary Objective: Improve the accuracy and reliability of the K-nearest neighbors (KNN) algorithm in classifying dataset instances by incorporating advanced techniques.
- Targeted Outcome: Enhance predictive performance and ensure robustness against class overlap

2. Outline of the approach

- Data Preparation: Utilized a comprehensive dataset from OpenML involving preprocessing steps such as encoding categorical variables and normalization to ensure data uniformity.
- Algorithm Enhancement:
  - Integration of Bagging: Implemented bagging and enhanced model stability by training multiple KNN models on various subsets of the data.
  - Adding weights: Implemented weights on the predict functions to favor near neighbors
- Evaluation Techniques:
  - Conducted rigorous testing using a split of training and test data to evaluate model performance.
  - Generated ROC curves and classification reports to assess accuracy and effectiveness.
  - And 2D plots to analyze the correctness of our changes in the algorithm

3. Summary of results

- Increased accuracy on some datasets
- More robust to class overlap

K-NEAREST NEIGHBORS (KNN)

What’s the KNN algorithm? And how does it work?

KNN is a simple, instance-based learning algorithm where the function is only approximated locally and all computation is deferred until classification.

It works by finding the k closest training examples in feature space and making predictions based on their labels.

KNN uses distance metrics to find the closest neighbors. If one feature has a broad range of values, it can dominate the distance measure compared to another feature with a narrower range.

Algorithm behavior regarding the data

As we said, KNN uses distance metrics to find the closest neighbors, this behavior implies the need for feature scaling (standardization or normalization) to ensure all features contribute equally to the distance calculations.

The k-nearest neighbors KNN algorithm is sensitive to class overlap due to its fundamental reliance on the proximity of examples in the feature space to make predictions.

The choice of k (the number of neighbors considered) is crucial in KNN. A small k can make the classifier too sensitive to noise, whereas a large k can include too many points from overlapping classes, diluting the influence of the nearest points.

MOTIVATION

Enhancing KNN with Bagging

Challenges in Traditional KNN:

- **Sensitivity to Noisy Data:** KNN can be overly sensitive to noise in the dataset, as it makes predictions based solely on the nearest neighbors.
- **Unequal Feature Influence:** Without standardization, features with larger ranges dominate the outcome, potentially skewing results.

Proposed Solution:

- Utilization of Bagging: To reduce variance and improve stability.
- Weighted Neighbors: To ensure that near neighbors have more weight in the decision of the object class that is being observed

Why Bagging?

Bagging is a technique that combats overfitting by introducing ensemble learning, where multiple models (each trained on a subset of the data) vote on the outcome This reduces the impact of outliers and noise by averaging multiple predictions, thereby improving overall model accuracy and robustness.

Weights

Weights are introduced at the level of individual distance calculations and are used to weight the influence of each neighbor based on their proximity. This method enhances the sensitivity of the KNN classifier to more relevant (closer) neighbors, theoretically improving its performance, especially in cases with noisy data or outliers.

Implementing KNN with Bagging

How Bagging Works with KNN:

- Bootstrap Aggregating: Multiple subsets of the original dataset are created with replacement, known as bootstrap samples.

- Training Multiple Models: A KNN model is trained on each of these samples. Unlike standard KNN, each model in the ensemble may use a different weighted distance function.

- Aggregation: For classification, the final prediction is made by majority voting among all models; for regression, it is the average of the predictions.

Expected benefits:

By applying these changes we expect the algorithm to improve it’s accuracy due to the fact that we aim to reduce noise and focus on crucial features.
Creates a bigger flexibility to diverse data characteristics
Becomes more robust to class overlap problems

DESCRIPTION OF THE ENHANCED MODEL

Implementing weights on KNN

```
# Calculate distances
distances = [distance_func(x, example.ravel()) for example in X]
# Avoid division by zero
distances = [max(d, 1e-5) for d in distances]
# Calculate weights inversely proportional to distances
weights = [1/d for d in distances]
```

HERE, EACH WEIGHT IS CALCULATED AS THE RECIPROCAL OF THE DISTANCE, ENSURING THAT CLOSER NEIGHBORS HAVE A HIGHER INFLUENCE ON THE OUTCOME. A MINIMUM DISTANCE VALUE (1E-5) IS USED TO PREVENT DIVISION BY ZERO.

THE FUNCTION ADDS THE WEIGHT TO A RUNNING TOTAL FOR EACH CLASS IN A DICTIONARY CLASS_WEIGHTS. THE CLASS WITH THE HIGHEST TOTAL WEIGHT IS SELECTED AS THE PREDICTION. THIS MEANS THAT NOT JUST THE PRESENCE OF A CLASS AMONG THE NEIGHBORS MATTERS, BUT ALSO HOW CLOSE THESE NEIGHBORS ARE TO THE QUERY POINT, WEIGHTED BY THEIR PROXIMITY.

```
def aggregate(self, neighbors):
    class_weights = {}
    for distance, target, weight in neighbors[:self.k]:
        if target in class_weights:
            class_weights[target] += weight
        else:
            class_weights[target] = weight
    return max(class_weights, key=class_weights.get)
```

Experimental Setup and Data Overview

Experimental Setup and Data Overview

Ti

1. **Experimental Setup:**
   - **Objective:** To test the efficacy of enhanced KNN algorithms with weighted distances and bagging.
   - **Environment:** Description of the software and hardware environment where tests were conducted (e.g., Python version, sklearn library, CPU/GPU specs).
2. **Datasets:**
   - **Source:** OpenML
   - **Characteristics:**
     - **Type of Data:** Classification
     - **Features:** Depends on the dataset, the data is mainly numeric as it was easier to work with
     - **Characteristics to look for:**
       - **Noise**
       - **Class Overlap**
       - **Focus on High Dimensionality:** Challenges with distance-based classification and curse of dimensionality
       - **Impact of Feature Scaling:** Importance of normalization in distance computation for KNN

Hyperparameters

Number of Neighbors (k):

- **Definition:** The number k represents the count of nearest neighbors to consider when making a prediction. It is a crucial parameter that balances the bias-variance tradeoff in KNN.
- **Impact on Performance:**
  - **Low k values:** Can lead to models that are highly sensitive to noise in the training data, potentially causing overfitting.
  - **High k values:** Tend to smooth out the prediction as the influence of the noise decreases, but can also obscure important local data patterns, leading to underfitting.

Distance Metrics:

- **Role:** The choice of distance metric dictates how similarity between instances is quantified. It fundamentally shapes the decision boundaries of the KNN algorithm.

Bagging Parameters:

- **Purpose of Bagging:** To reduce variance and avoid overfitting by training multiple KNN models on various bootstrap samples of the training data and then averaging their predictions.
- **Key Parameters:**
  - **Number of Estimators (n_estimators):** Refers to the number of individual KNN models to train.
  - **Bootstrap Sampling:** Whether to sample with replacement from the training dataset.
  - **Max Features:** Determines the subset of features to use for training each model in the ensemble.

DATA ANALYSIS

HOW OUR CHANGES BEHAVE ON TESTED DATASETS

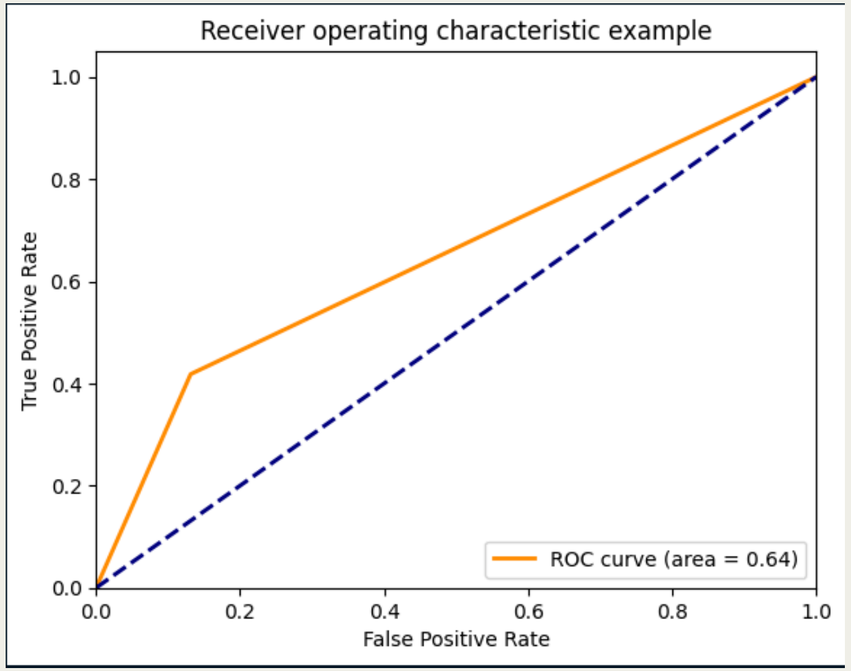
Performance Analysis and Results

Performance Estimation Methodology:

- **Validation Technique:** Use of train-test split
- **Metrics Used:** Accuracy, Precision, Recall, F1-Score, ROC-AUC.

Presentation of Results:

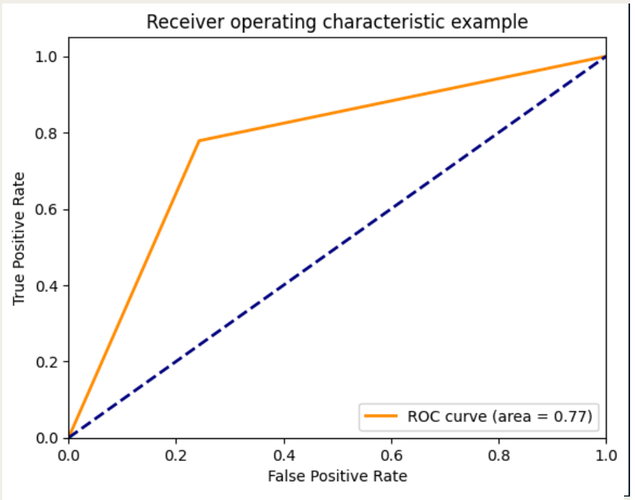
- **Visuals:** Include ROC curves, confusion matrices, and accuracy graphs.
- **Quantitative Findings:** Best performance metrics obtained.

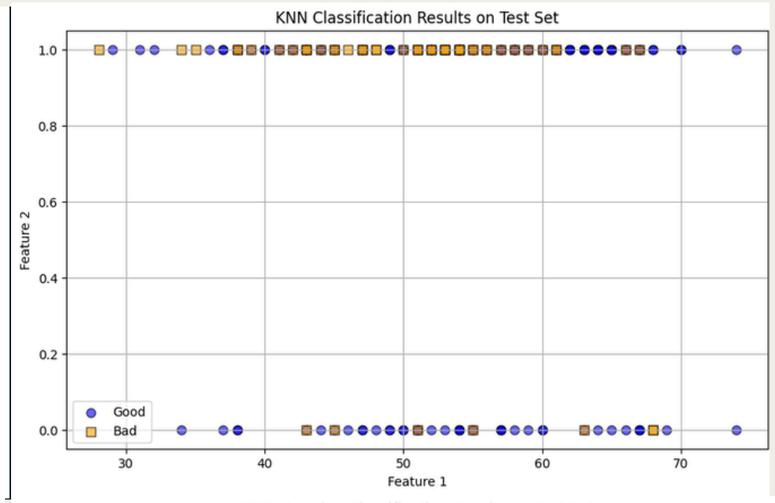


RESULTS COMPARISON

As mentioned earlier various k's where tested and the best k is chosen.
The data was split on 70% training set and 30% testing set

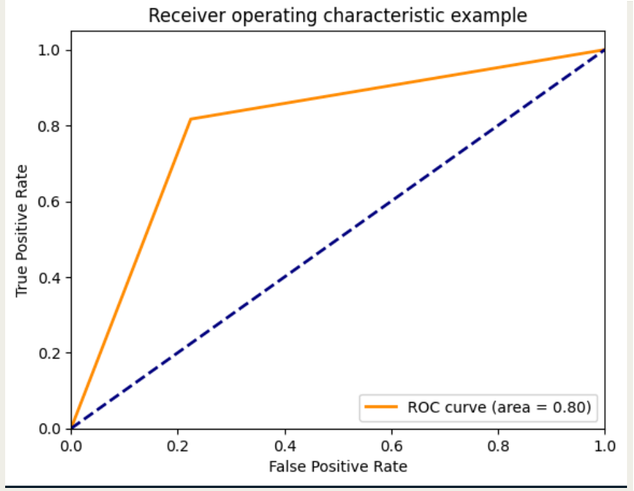
KNN BASE

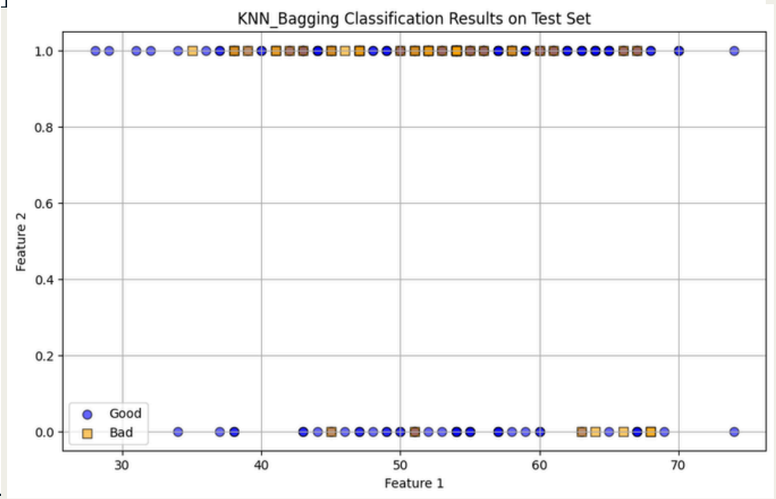




	precision	recall	f1-score	support
0	0.74	0.76	0.75	107
1	0.80	0.78	0.79	131
accuracy			0.77	238
macro avg	0.77	0.77	0.77	238
weighted avg	0.77	0.77	0.77	238

KNN BAGGING





	precision	recall	f1-score	support
0	0.82	0.75	0.78	107
1	0.81	0.86	0.83	131
accuracy			0.81	238
macro avg	0.81	0.81	0.81	238
weighted avg	0.81	0.81	0.81	238

RESULTS COMPARISON - HOW DID THE ACCURACY INCREASE?

1. Diversity Through Multiple Distance Functions

Using different distance functions within the ensemble of KNN models introduces diversity in how each model perceives similarity between data points. Each distance metric has strengths in different aspects or types of data

2. Robustness Through Bagging

- Each individual KNN model in the ensemble may have different weaknesses, but when combined, their uncorrelated errors can cancel out, leading to more robust predictions.
- Bagging allows each model to make predictions independently on randomly resampled data subsets, increasing the diversity of the data scenarios they are trained on, which can improve the generalization ability of the ensemble.

3. Weighted Aggregation

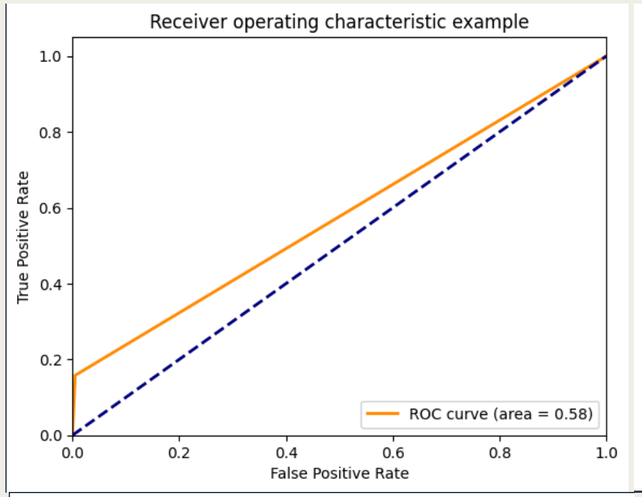
Our implementation uses a weighted aggregation method where weights are inversely proportional to the distances. This means that nearer neighbors contribute more to the final class decision than farther ones. This focus on nearer neighbors can significantly enhance accuracy because closer points are typically more similar and hence more relevant in many scenarios.

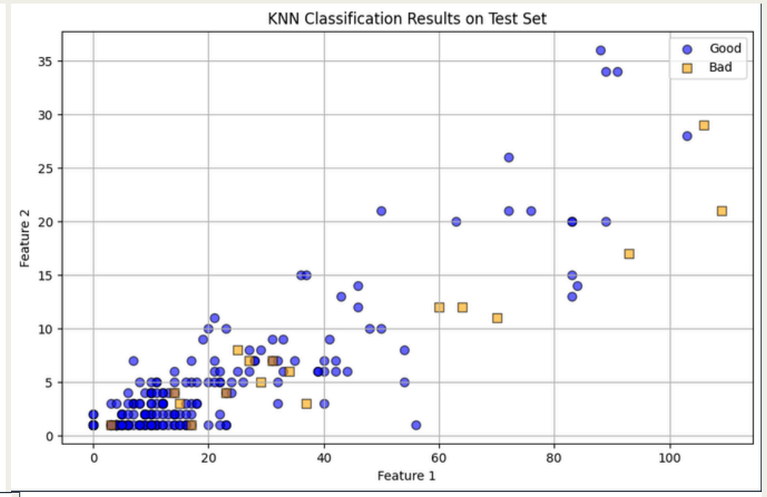
4. Majority Voting Across Models

The final prediction step in your KNN Bagging approach uses majority voting to decide the output class.

RESULTS COMPARISON - Decrease on accuracy

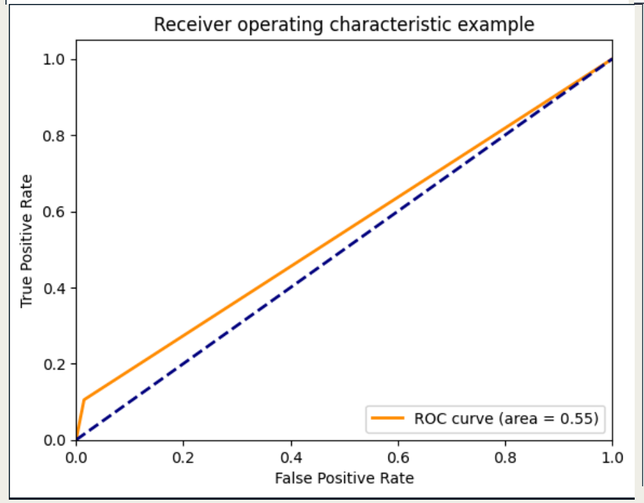
KNN BASE

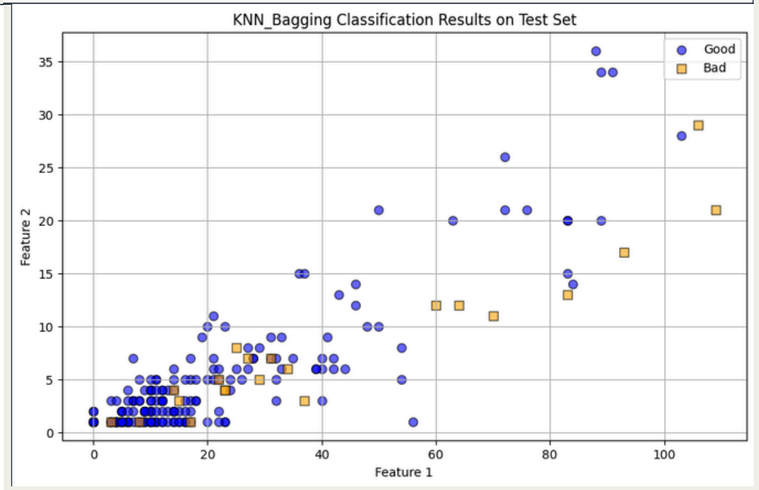




	precision	recall	f1-score	support
0	0.93	1.00	0.96	203
1	0.75	0.16	0.26	19
accuracy			0.92	222
macro avg	0.84	0.58	0.61	222
weighted avg	0.91	0.92	0.90	222

KNN BAGGING





	precision	recall	f1-score	support
0	0.92	0.99	0.95	203
1	0.40	0.11	0.17	19
accuracy			0.91	222
macro avg	0.66	0.55	0.56	222
weighted avg	0.88	0.91	0.89	222

RESULTS COMPARISON - WHY DID THE ACCURACY DECREASE?

The decreasing of the accuracy on a tested dataset can be due to various reasons, but we tried to narrow it down to the biggest causes for that.

One of the causes may be that there is a mismatch between the distance functions and some data characteristics of our dataset

- **Inappropriate Metrics:** If the data has certain features that are more important but are being underweighted or ignored by the selected distance functions, the model may perform poorly.
- **Scale Sensitivity:** Some metrics like Euclidean or Minkowski are sensitive to the scale of the data. Without proper normalization, these metrics might not perform well, especially if some features dominate because of their scale.

Improper Weighting Scheme

The approach of weighting inversely proportional to distance assumes that closer neighbors are always more relevant, which might not always be true, especially in datasets with overlapping classes or non-uniform density:

- **Outlier Influence:** Weights might amplify the influence of outliers or noisy data points if these points are close to the query point.
- **Distance Weight Overemphasis:** If weights are calculated improperly, they might distort the contribution of genuinely significant neighbors.

Hypothesis study and final remarks

As mentioned in the beginning of the presentation, we tried to enhance our KNN by using bagging and adding weights to our neighbors, thus reducing class overlap problems and noise.

Looking at our results, not every dataset improved although, the ones that didn’t, their accuracy wasn’t affected significatively.

Our result could have been better if the datasets chosen where more fitting to the algorithm problems.

We also observed that changing the hyperparameters didn’t influence the accuracy of the datasets.

THANK YOU!