Project 2: Feature Selection with Nearest Neighbor

Student Name1: Ryan Noghani SID: 862381739 Lecture Session: 1

Student Name2: Gabriel Serrano SID: 862261377 LectureSession:1

Student Name3: Armando Hernandez SID: 862323898 Lecture Session:1

Student Name4: Michael Zheng SID: 862265463 Lecture Session: 1

Student Name5: Kirtana Venkat SID: 862284458 Lecture Session: 1

Solution: Datasets 31

Dataset	Best Feature Set	Accuracy
Small Number: <insert your small dataset number>	Forward Selection = {3, 4 ,5}	0.89
	Backward Elimination = {3, 5 7}	0.78
	Custom Algorithm = Not implemented	0.91
Large Number: <insert your large dataset number>	Forward Selection = {15, 4 ,1}	0.81
	Backward Elimination = {15, 16, 7}	0.92
	Custom Algorithm = {15, 4, 3}	0.87

----------<BeginReport>----------

In completing this project, I consulted following resources:

Stack overflow, geeksforgeeks, numpy documentation, python documentation.

Contribution of each student in the group:

1. Armando: Implemented the Nearest_Neighbor function and added comments to all the functions for easy understanding of the code;
2. Ryan: Implemented the Evaluation_Function, and Backwards_Selection() functions, and tested the program on the dataset;
3. Gabriel: Tested the program on the large dataset, did some debugging, contributed partially to the forward search function, and worked on the I-VI portion of the report;
4. Michael: Implemented Forward Search, tested it on the small dataset and implemented the UI for the project;
5. Kirtana: Implemented the Backward_Selection() function, did the remaining portion of the report, and added numpy as np as well.

I. Introduction

This project explores the implementation and application of the nearest neighbor algorithm. The nearest neighbor program works by taking one data point, calculating its distance from all the other points, and then classifying said original point as the class of its nearest neighbor. This algorithm is undoubtedly powerful. However, it can be significantly impacted by the presence of irrelevant features, which can degrade its performance.

The purpose of this project is to implement a feature selection process that prioritizes the most relevant features. We use various algorithms in order to select the most relevant features that will ultimately allow the algorithm to be used to its full effectiveness.

II. Challenges

- Putting all the parts of the algorithm together.
- Properly normalizing the data to perform leave-one-out cross-validation
    - Properly handling matrix operations during normalization
- Connecting the nearest neighbor classifier to our feature search algorithms
    - Use it to calculate the accuracy of the given feature set in the evaluation function
- Making sure that forward/backing selections are efficient and make sure correct methods are being implemented

III. Code Design

Our code is modular. Every function does one thing.

IV. Dataset details

The General Small Dataset: 10 features, 100 instances

The General Large Dataset: 40 features, 1000 instances

CS170_Spring_2024_Small_data__31.txt: 10 features, 100 instances

CS170_Spring_2024_Large_data__31.txt: 40 features,1000 instances

Plot some features and color code them by class and explore your dataset.

V. Algorithms

1. Forward Selection: A feature selection method used to identify the most relevant features for a predictive model by iteratively adding features to the model and evaluating performance.

- Start with no features:
  - Begin with an empty set of features
- Evaluate Each Feature:
  - For each feature not yet included, temporarily add it to the current set of features.
  - Evaluate the performance of the model with this feature set(e.g., using cross-validation).
- Select Best Feature:
  - Choose the feature that improves the model's performance the most.
- Update Feature Set:
  - Add the selected feature to the current set of features.
- Repeat:
  - Repeat steps 2-4 until no further improvement is observed or all features have been considered.

2. Backward Elimination: A feature selection method that starts with all features and iteratively removes the least significant features to enhance model performance.

- Start with all features:
  - Begin with the full set of features.
- Evaluate each feature:
  - For each feature in the current set, temporarily remove it and evaluate the model's performance.
- Select least important feature:
  - Identify the feature whose removal results in the least decrease(or highest increase) in model performance;
- Update feature set:
  - Remove the selected feature from the current set.

- Repeat:
    - Repeat steps 2-4 until no further improvement is observed or only a predetermined number of features remain.

3. Your custom algorithm (optional)

<explain each algorithm in brief>

VI. Analysis

Experiment 1: Comparing Forward Selection vs Backward Elimination.

<do this experiment for sure>

Compare accuracy with no feature selection vs with feature selection.

No features:

The General Small Dataset (FS: 75%, BE: 75%)

The General Large Dataset(FS: 81%, BE: 81%)

CS170_Spring_2024_Small_data__31.txt (FS: 79%, BE: 79%)

CS170_Spring_2024_Large_data__31.txt(FS: 83.5%, BE: 83.5%)

All features:

Compare feature set and accuracy for forward selection vs backward elimination.

The General Small Dataset(10 features being as input):

Forward selection: 92% {3, 5}

Backward elimination: 82% {2, 4, 5, 7, 10}

The General Large Dataset(40 features being as input):

Forward selection: 95.5% {1, 27}

Backward elimination: 72.2% {2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40}

CS170_Spring_2024_Small_data__31.txt (10 features being as input):

Forward selection: 98% {9,1}

Backward elimination: 98% {9,1}

CS170_Spring_2024_Large_data__31.txt(40 features being as input):

Forward selection: 83.5%: {no features since random evaluation gives the

highest accuracy }

Backward elimination: 77.1%: {1, 2, 3, 4, 6, 8, 9, 10, 11, 12, 13, 15, 16, 18, 20, 21, 22, 23, 25, 26, 27, 28, 29, 30, 31, 32, 33, 35, 37, 38, 39, 40}

- Forward Selection
  - Pros
    - Efficient for smaller sets
    - It is easier to identify what features improve accuracy
  - Cons
    - For larger datasets, evaluating each potential feature to add can be expensive
    - Makes locally optimal choices and may not make globally optimal solution
- Backward selection
  - Pros
    - Creates a model that includes potential feature interactions from the beginning.
    - Help in identifying and eliminating redundant or irrelevant features.
  - Cons
    - Produces lower accuracy score
    - It is expensive to start with all the features
    - Might be more complex to implement and understand than forward selection
    - Makes locally optimal choices and may not make globally optimal solution

Experiment 2: Effect of normalization

Compare accuracy when using normalized vs unnormalized data.

Experiment 3: Effect of number neighbors (k)

Plot accuracy vs increasing values of k and examine the trend.

VII. Conclusion

The purpose of creating this code was to help us understand the nearest neighbor algorithm and its correlation with features. By implementing forward and backward selection, we sought to understand how various combinations of features aided or deterred the accuracy of our algorithm. Overall, this project demonstrated the importance of feature selection in enhancing the accuracy and efficiency of machine learning models.

VIII. Trace of your small dataset <paste the trace of your small dataset here>

CS170_Spring_2024_Small_data__31 trace (with 10 features as input):

Using forward selection:

```
Please enter total number of features: 10

Type the number of the algorithm you want to run.


        Forward Selection
        Backward Elimination
        Bertie's Special Algorithm

1
This dataset has 10 features (not including the class attribute), with 100 instances.
Using no features and "random" evaluation, I get an accuracy of 79.0%

Beginning search

        Using feature(s) {1} accuracy is 74.0%

        Using feature(s) {2} accuracy is 62.0%

        Using feature(s) {3} accuracy is 70.0%

        Using feature(s) {4} accuracy is 65.0%

        Using feature(s) {5} accuracy is 66.0%

        Using feature(s) {6} accuracy is 69.0%

        Using feature(s) {7} accuracy is 66.0%

        Using feature(s) {8} accuracy is 63.0%

        Using feature(s) {9} accuracy is 80.0%

        Using feature(s) {10} accuracy is 71.0%

Feature set {9} was best, accuracy is 80.0%

        Using feature(s) {9, 1} accuracy is 98.0%

        Using feature(s) {9, 2} accuracy is 88.0%

        Using feature(s) {9, 3} accuracy is 88.0%

        Using feature(s) {9, 4} accuracy is 81.0%

        Using feature(s) {9, 5} accuracy is 88.0%

        Using feature(s) {9, 6} accuracy is 83.0%

        Using feature(s) {9, 7} accuracy is 85.0%

        Using feature(s) {8, 9} accuracy is 80.0%

        Using feature(s) {9, 10} accuracy is 75.0%

Feature set {9, 1} was best, accuracy is 98.0%

        Using feature(s) {9, 2, 1} accuracy is 88.0%

        Using feature(s) {9, 3, 1} accuracy is 90.0%

        Using feature(s) {9, 4, 1} accuracy is 88.0%

        Using feature(s) {9, 5, 1} accuracy is 89.0%

        Using feature(s) {9, 1, 6} accuracy is 92.0%

        Using feature(s) {9, 1, 7} accuracy is 87.0%

        Using feature(s) {8, 9, 1} accuracy is 90.0%

        Using feature(s) {9, 10, 1} accuracy is 94.0%

(Warning, Accuracy has decreased!)
Finished search!! The best feature subset is,  {9, 1} which has an accuracy of 98.0%
```

Using backward elimination:

```
Please enter total number of features: 10

Type the number of the algorithm you want to run.


        Forward Selection
        Backward Elimination
        Bertie's Special Algorithm

2
This dataset has 10 features (not including the class attribute), with 100 instances.
Using all features and "random" evaluation, I get an accuracy of 68.0%

Beginning search

        Using feature(s) {2, 3, 4, 5, 6, 7, 8, 9, 10} accuracy is 66.0%

        Using feature(s) {1, 3, 4, 5, 6, 7, 8, 9, 10} accuracy is 67.0%

        Using feature(s) {1, 2, 4, 5, 6, 7, 8, 9, 10} accuracy is 71.0%

        Using feature(s) {1, 2, 3, 5, 6, 7, 8, 9, 10} accuracy is 73.0%

        Using feature(s) {1, 2, 3, 4, 6, 7, 8, 9, 10} accuracy is 73.0%

        Using feature(s) {1, 2, 3, 4, 5, 7, 8, 9, 10} accuracy is 68.0%

        Using feature(s) {1, 2, 3, 4, 5, 6, 8, 9, 10} accuracy is 64.0%

        Using feature(s) {1, 2, 3, 4, 5, 6, 7, 9, 10} accuracy is 67.0%

        Using feature(s) {1, 2, 3, 4, 5, 6, 7, 8, 10} accuracy is 70.0%

        Using feature(s) {1, 2, 3, 4, 5, 6, 7, 8, 9} accuracy is 73.0%

Feature set {1, 2, 3, 5, 6, 7, 8, 9, 10} was best, accuracy is 73.0%

        Using feature(s) {2, 3, 5, 6, 7, 8, 9, 10} accuracy is 62.0%

        Using feature(s) {1, 3, 5, 6, 7, 8, 9, 10} accuracy is 74.0%

        Using feature(s) {1, 2, 5, 6, 7, 8, 9, 10} accuracy is 76.0%

        Using feature(s) {1, 2, 3, 6, 7, 8, 9, 10} accuracy is 72.0%

        Using feature(s) {1, 2, 3, 5, 7, 8, 9, 10} accuracy is 77.0%

        Using feature(s) {1, 2, 3, 5, 6, 8, 9, 10} accuracy is 74.0%

        Using feature(s) {1, 2, 3, 5, 6, 7, 9, 10} accuracy is 71.0%

        Using feature(s) {1, 2, 3, 5, 6, 7, 8, 10} accuracy is 72.0%
```

```
        Using feature(s) {1, 2, 3, 5, 6, 7, 8, 10} accuracy is 72.0%
        Using feature(s) {1, 2, 3, 5, 6, 7, 8, 9} accuracy is 71.0%
Feature set {1, 2, 3, 5, 7, 8, 9, 10} was best, accuracy is 77.0%
        Using feature(s) {2, 3, 5, 7, 8, 9, 10} accuracy is 68.0%
        Using feature(s) {1, 3, 5, 7, 8, 9, 10} accuracy is 80.0%
        Using feature(s) {1, 2, 5, 7, 8, 9, 10} accuracy is 79.0%
        Using feature(s) {1, 2, 3, 7, 8, 9, 10} accuracy is 79.0%
        Using feature(s) {1, 2, 3, 5, 8, 9, 10} accuracy is 80.0%
        Using feature(s) {1, 2, 3, 5, 7, 9, 10} accuracy is 82.0%
        Using feature(s) {1, 2, 3, 5, 7, 8, 10} accuracy is 70.0%
        Using feature(s) {1, 2, 3, 5, 7, 8, 9} accuracy is 80.0%
Feature set {1, 2, 3, 5, 7, 9, 10} was best, accuracy is 82.0%
        Using feature(s) {2, 3, 5, 7, 9, 10} accuracy is 70.0%
        Using feature(s) {1, 3, 5, 7, 9, 10} accuracy is 84.0%
        Using feature(s) {1, 2, 5, 7, 9, 10} accuracy is 83.0%
        Using feature(s) {1, 2, 3, 7, 9, 10} accuracy is 84.0%
        Using feature(s) {1, 2, 3, 5, 9, 10} accuracy is 83.0%
        Using feature(s) {1, 2, 3, 5, 7, 10} accuracy is 67.0%
        Using feature(s) {1, 2, 3, 5, 7, 9} accuracy is 78.0%
Feature set {1, 3, 5, 7, 9, 10} was best, accuracy is 84.0%
        Using feature(s) {3, 5, 7, 9, 10} accuracy is 76.0%
        Using feature(s) {1, 5, 7, 9, 10} accuracy is 88.0%
        Using feature(s) {1, 3, 7, 9, 10} accuracy is 83.0%
        Using feature(s) {1, 3, 5, 9, 10} accuracy is 80.0%
        Using feature(s) {1, 3, 5, 7, 10} accuracy is 73.0%
        Using feature(s) {1, 3, 5, 7, 9} accuracy is 79.0%
```

```
Feature set {1, 5, 7, 9, 10} was best, accuracy is 88.0%

        Using feature(s) {9, 10, 5, 7} accuracy is 82.0%

        Using feature(s) {1, 10, 9, 7} accuracy is 91.0%

        Using feature(s) {1, 10, 5, 9} accuracy is 80.0%

        Using feature(s) {1, 10, 5, 7} accuracy is 76.0%

        Using feature(s) {1, 5, 9, 7} accuracy is 89.0%

Feature set {1, 7, 9, 10} was best, accuracy is 91.0%

        Using feature(s) {9, 10, 7} accuracy is 78.0%

        Using feature(s) {1, 10, 9} accuracy is 94.0%

        Using feature(s) {1, 10, 7} accuracy is 73.0%

        Using feature(s) {1, 9, 7} accuracy is 87.0%

Feature set {1, 9, 10} was best, accuracy is 94.0%

        Using feature(s) {9, 10} accuracy is 75.0%

        Using feature(s) {1, 10} accuracy is 74.0%

        Using feature(s) {1, 9} accuracy is 98.0%

Feature set {1, 9} was best, accuracy is 98.0%

        Using feature(s) {9} accuracy is 80.0%

        Using feature(s) {1} accuracy is 74.0%

(Warning, Accuracy has decreased!)
Finished search!! The best feature subset is,  {1, 9} which has an accuracy of 98.0%
```