Project 2 CS205: Introduction to Artificial Intelligence, Dr. Eamonn Keogh

Pei-Yun Jan
SID: 862548790
Email: pjan002@ucr.edu
Date: June-07-2025

In completing this project, I consulted information from the following sources.

- Wikipedia. K nearest neighbor: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
- Geeksforgeeks. K nearest neighbor: https://www.geeksforgeeks.org/k-nearest-neighbors-with-python-ml/, https://www.geeksforgeeks.org/k-nearest-neighbours/
- All the slides from professor Eamonn Keogh's lecture

My code at Github:
https://github.com/peiyunjan0807/CS205Project2_FeatureSelectionWithNearestNeighbor

Outline of this report:

- Cover page: (this page)
- My report, pages 2 to 5.
- Part 1, page 2 to 4.
- Part 2, page 5.
- Output result using forward selection on the small dataset, page 6 to 7.
- Output result using backward elimination on the small dataset, page 8 to 9.

CS205: Project 2: Feature Selection with Nearest Neighbor

Pei-Yun Jan, SID 862548790, June-07-2025

Part 1

In this project, we applied two classic feature selection methods Forward Selection and Backward Elimination on multiple datasets to evaluate classification accuracy. The goal was to identify the most informative subsets of features that lead to the best classification performance. Results are reported visually and analyzed in terms of accuracy trends.

In Figure 1 displays the performance of Forward Selection on the small dataset CS205_small_Data__25.txt.

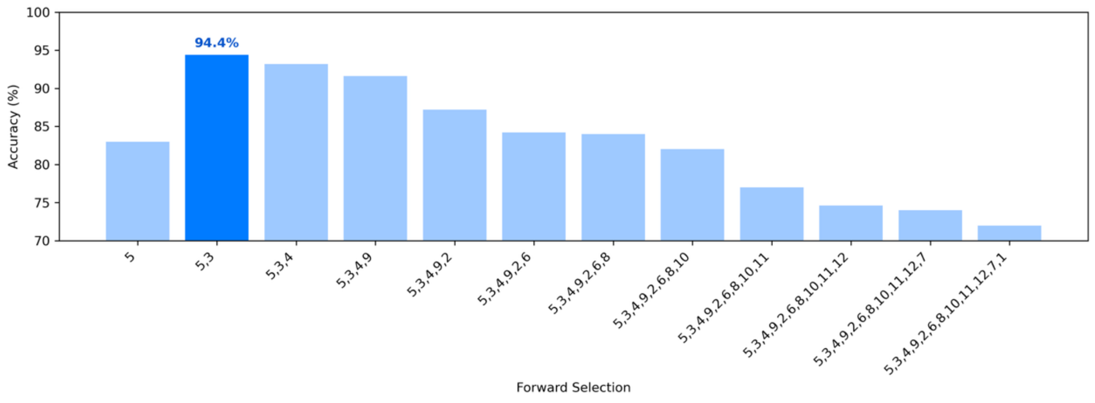

Figure 1: Forward Selection on Small Dataset 25

From the plot, we observe that the accuracy improves substantially at the beginning of the selection process. Adding features 5 and 3 achieves the highest accuracy of 94.4%, indicating that these two features alone are highly informative. Adding more features beyond this point leads to a decline in performance, likely due to noise or redundancy.

In Figure 2 shows the result of Backward Elimination on the same dataset.

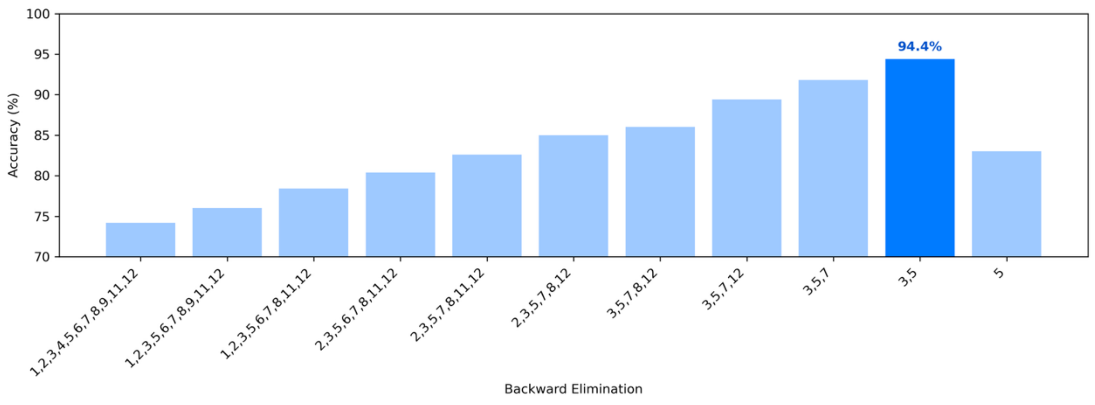

Figure 2: Backward Elimination on Small Dataset 25

Interestingly, the backward search also identifies the {3,5} subset as the best, achieving 94.4% accuracy same as Forward Selection. This consistency reinforces the conclusion that features 3 and 5 are the most useful.

Conclusion for Small Dataset: Both methods converged on {3,5} as the optimal subset. We can be confident these features are most predictive, and using only them may avoid overfitting.

Next, focused on the larger and more challenging dataset assigned to me. In Figure 3 depicts the Forward Selection results on the large dataset CS205_large_Data__14.

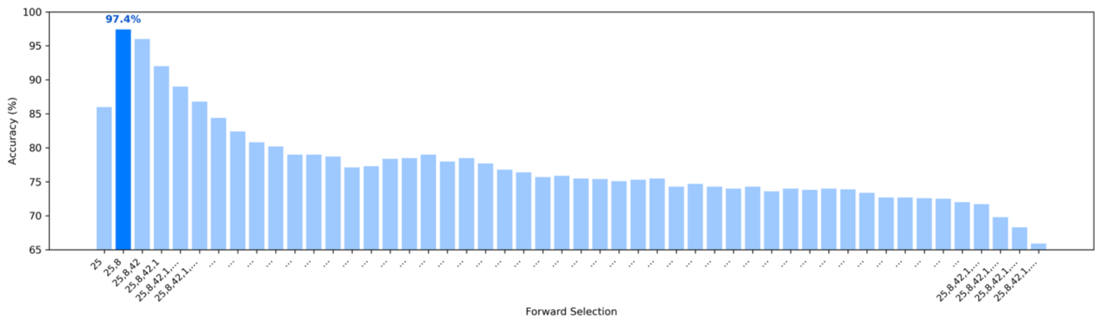

Figure 3: Forward Selection on Large Dataset 14

The initial inclusion of feature 25 gives a decent starting accuracy. When combined with feature 8, accuracy spikes to a remarkable 97.4%. However, adding more features beyond this causes a steady decline. This trend suggests 25 and 8 provide nearly all the predictive power, and others add noise or overfit the model.

In Figure 4 shows the results of applying Backward Elimination to the same large dataset.

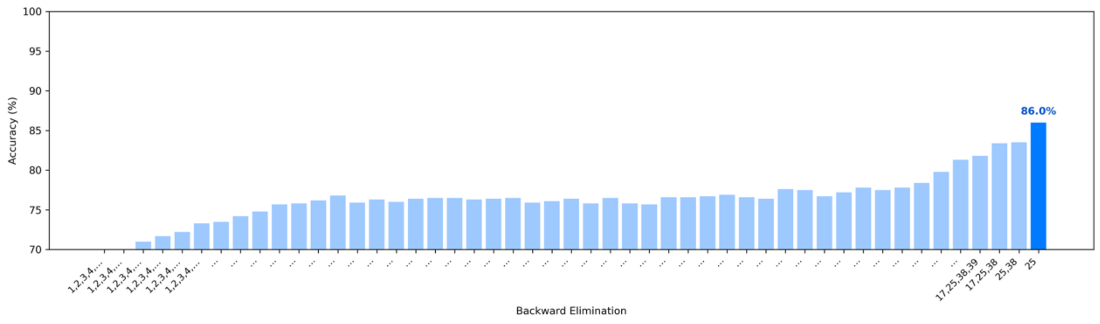

Figure 4: Backward Elimination on Large Dataset 14

Here, we begin with all features and progressively eliminate the least useful. The final subset {25} achieves an accuracy of 86.0%. While not as high as Forward Selection's best result, the difference in trends confirms feature 25 is fundamentally important.

Conclusion for Large Dataset: Feature 25 is a standout, and adding 8 further enhances performance. Forward Selection outperformed Backward Elimination here, likely because good features were added early before irrelevant ones interfered.

Computational effort for search: All feature selection algorithms were implemented in C++ and executed on a MacBook Air with an Intel Core i5 processor, 8GB RAM.

The table 1 below summarizes the runtime performance.

	Small dataset (12 features,500 instances)	Large dataset (50 features,1000instances)
Forward Selection	6.2 seconds	31.45 minutes
Backward Elimination	8.8 seconds	44.20 minutes

Table 1: Comparison of Feature Selection Runtime

On the small dataset (12 features, 500 instances), Forward Selection completed in 6.2 seconds, identifying {5, 3} as the best subset with an accuracy of 94.4%. Backward Elimination took slightly longer at 8.8 seconds, as it evaluates all feature removal combinations, but ultimately converged on the same optimal subset {3, 5}. On the large dataset (50 features, 1000 instances), the computational cost increased significantly. Forward Selection took 31.45 minutes, incrementally building toward the best performing subset {25, 8} with 97.4% accuracy. In contrast, Backward Elimination required 44.20 minutes and selected {25} as the best individual feature, yielding a lower accuracy of 86.0%. These results highlight a clear tradeoff between accuracy and computational efficiency: although Forward Selection tends to be more time consuming, it often identifies higher performing feature subsets, especially in high dimensional spaces.

Part 2

I selected a real-world dataset from the UCI Machine Learning Repository. The dataset I chose is the Wine Dataset[a], which contains 13 continuous (real valued) features and classifies wines into 3 categories based on chemical analysis of cultivars.

In Figure 5 presents the accuracy of the nearest neighbor classifier as more features were added during forward selection.

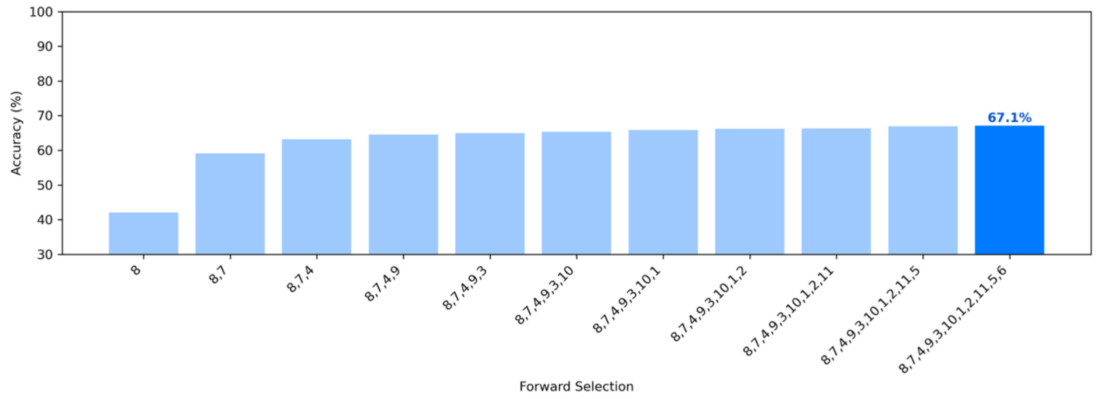

Figure 5: Forward Selection on Wine Dataset

At the start, using only feature 8 gave a low accuracy of 42.4%. However, as the algorithm added more features, performance improved steadily. The best accuracy was achieved using the full subset {8, 7, 4, 9, 3, 10, 1, 2, 11, 5, 6}, which resulted in 67.1% accuracy.

The first features added 8, 7, and 4 appear to be the most informative and helpful. Feature 8 corresponds to proline, a compound associated with grape variety and fermentation processes[b]. Feature 7 is flavanoids, which are known to vary significantly among wine types and affect taste and color[c]. Feature 4, alcalinity of ash, relates to mineral composition and may indirectly reflect the soil and cultivar differences[d]. These features likely capture core chemical properties that differentiate wine types.

[a] https://archive.ics.uci.edu/ml/datasets/wine
[b] https://en.wikipedia.org/wiki/Proline
[c] https://en.wikipedia.org/wiki/Wine_chemistry
[d] Ribéreau-Gayon, P., Glories, Y., Maujean, A., & Dubourdieu, D. (2006). Handbook of Enology, Volume 2: The Chemistry of Wine – Stabilization and Treatments. John Wiley & Sons.

Below I show a single trace of my algorithm. I am *only* showing Forward Selection on the small dataset.

```
Welcome to Bertie Woosters Feature Selection Algorithm.
Type in the name of the file to test : CS205_small_Data__25.txt

Type the number of the algorithm you want to run.
   1)  Forward Selection
   2)  Backward Elimination

1

This dataset has 12 features (not including the class attribute),
with 500 instances.

Running nearest neighbor with all 12 features, using "leaving-one-
out" evaluation, I get an accuracy of 72.0%

Beginning search.

   Using feature(s) {1} accuracy is 70.8%
   Using feature(s) {2} accuracy is 76.2%
   Using feature(s) {3} accuracy is 74.4%
   Using feature(s) {4} accuracy is 70.0%
   Using feature(s) {5} accuracy is 83.0%
   Using feature(s) {6} accuracy is 68.2%
   Using feature(s) {7} accuracy is 73.2%
   Using feature(s) {8} accuracy is 73.2%
   Using feature(s) {9} accuracy is 70.4%
   Using feature(s) {10} accuracy is 74.4%
   Using feature(s) {11} accuracy is 71.2%
   Using feature(s) {12} accuracy is 69.4%

Feature set {5} was best, accuracy is 83.0%

   Using feature(s) {5, 1} accuracy is 83.4%
   Using feature(s) {5, 2} accuracy is 81.6%
   Using feature(s) {5, 3} accuracy is 94.4%
   Using feature(s) {5, 4} accuracy is 82.4%
   Using feature(s) {5, 6} accuracy is 81.0%
   Using feature(s) {5, 7} accuracy is 84.6%
   Using feature(s) {5, 8} accuracy is 80.8%
   Using feature(s) {5, 9} accuracy is 84.4%
   Using feature(s) {5, 10} accuracy is 85.2%
   Using feature(s) {5, 11} accuracy is 84.4%
   Using feature(s) {5, 12} accuracy is 81.6%

Feature set {5, 3} was best, accuracy is 94.4%

   Using feature(s) {5, 3, 1} accuracy is 91.4%
   Using feature(s) {5, 3, 2} accuracy is 90.6%
   Using feature(s) {5, 3, 4} accuracy is 93.2%
   Using feature(s) {5, 3, 6} accuracy is 91.8%
   Using feature(s) {5, 3, 7} accuracy is 91.8%
   Using feature(s) {5, 3, 8} accuracy is 90.8%
   Using feature(s) {5, 3, 9} accuracy is 93.2%
   Using feature(s) {5, 3, 10} accuracy is 93.0%
   Using feature(s) {5, 3, 11} accuracy is 90.0%
   Using feature(s) {5, 3, 12} accuracy is 91.2%
```

```
(WARNING, Accuracy has decreased! Continuing search in case of local
maxima)
Feature set {5, 3, 4} was best, accuracy is 93.2%

   Using feature(s) {5, 3, 4, 1} accuracy is 88.4%
   Using feature(s) {5, 3, 4, 2} accuracy is 89.4%
   Using feature(s) {5, 3, 4, 6} accuracy is 87.2%
   Using feature(s) {5, 3, 4, 7} accuracy is 90.6%
   Using feature(s) {5, 3, 4, 8} accuracy is 88.0%
   Using feature(s) {5, 3, 4, 9} accuracy is 91.6%
   Using feature(s) {5, 3, 4, 10} accuracy is 88.8%
   Using feature(s) {5, 3, 4, 11} accuracy is 89.0%
   Using feature(s) {5, 3, 4, 12} accuracy is 90.2%

::::            //I deleted about 2 pages of the trace to save space


(WARNING, Accuracy has decreased! Continuing search in case of local
maxima)
Feature set {5, 3, 4, 9, 2, 6, 8, 10, 11, 12, 7} was best, accuracy
is 74.0%

   Using feature(s) {5, 3, 4, 9, 2, 6, 8, 10, 11, 12, 7, 1} accuracy
is 72.0%

(WARNING, Accuracy has decreased! Continuing search in case of local
maxima)
Feature set {5, 3, 4, 9, 2, 6, 8, 10, 11, 12, 7, 1} was best,
accuracy is 72.0%

Finished search!! The best feature subset is {5, 3}, which has
accuracy of 94.4%
```

Below I show a single trace of my algorithm. I am showing Backward Elimination on the small dataset.

```
Welcome to Bertie Woosters Feature Selection Algorithm.
Type in the name of the file to test : CS205_small_Data__25.txt

Type the number of the algorithm you want to run.
    1)  Forward Selection
    2)  Backward Elimination

2

This dataset has 12 features (not including the class attribute),
with 500 instances.

Running nearest neighbor with all 12 features, using "leaving-one-
out" evaluation, I get an accuracy of 72.0%

Beginning search.

    Using feature(s) {2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12} accuracy is
74.0%
    Using feature(s) {1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12} accuracy is
73.4%
    Using feature(s) {1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12} accuracy is
71.6%
    Using feature(s) {1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12} accuracy is
73.8%
    Using feature(s) {1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12} accuracy is
69.2%
    Using feature(s) {1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12} accuracy is
72.6%
    Using feature(s) {1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12} accuracy is
73.0%
    Using feature(s) {1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12} accuracy is
71.6%
    Using feature(s) {1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12} accuracy is
71.8%
    Using feature(s) {1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12} accuracy is
74.2%
    Using feature(s) {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12} accuracy is
72.0%
    Using feature(s) {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11} accuracy is
69.8%
Feature set {1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12} was best, accuracy
is 74.2%

    Using feature(s) {2, 3, 4, 5, 6, 7, 8, 9, 11, 12} accuracy is
74.8%
    Using feature(s) {1, 3, 4, 5, 6, 7, 8, 9, 11, 12} accuracy is
73.2%
    Using feature(s) {1, 2, 4, 5, 6, 7, 8, 9, 11, 12} accuracy is
71.6%
    Using feature(s) {1, 2, 3, 5, 6, 7, 8, 9, 11, 12} accuracy is
76.0%
    Using feature(s) {1, 2, 3, 4, 6, 7, 8, 9, 11, 12} accuracy is
73.0%
    Using feature(s) {1, 2, 3, 4, 5, 7, 8, 9, 11, 12} accuracy is
75.0%
```

```
   Using feature(s) {1, 2, 3, 4, 5, 6, 8, 9, 11, 12} accuracy is
74.4%
   Using feature(s) {1, 2, 3, 4, 5, 6, 7, 9, 11, 12} accuracy is
75.4%
   Using feature(s) {1, 2, 3, 4, 5, 6, 7, 8, 11, 12} accuracy is
76.0%
   Using feature(s) {1, 2, 3, 4, 5, 6, 7, 8, 9, 12} accuracy is
74.6%
   Using feature(s) {1, 2, 3, 4, 5, 6, 7, 8, 9, 11} accuracy is
73.2%
Feature set {1, 2, 3, 5, 6, 7, 8, 9, 11, 12} was best, accuracy is
76.0%

   Using feature(s) {2, 3, 5, 6, 7, 8, 9, 11, 12} accuracy is 77.4%
   Using feature(s) {1, 3, 5, 6, 7, 8, 9, 11, 12} accuracy is 76.2%
   Using feature(s) {1, 2, 5, 6, 7, 8, 9, 11, 12} accuracy is 72.8%
   Using feature(s) {1, 2, 3, 6, 7, 8, 9, 11, 12} accuracy is 69.6%
   Using feature(s) {1, 2, 3, 5, 7, 8, 9, 11, 12} accuracy is 75.8%
   Using feature(s) {1, 2, 3, 5, 6, 8, 9, 11, 12} accuracy is 76.6%
   Using feature(s) {1, 2, 3, 5, 6, 7, 9, 11, 12} accuracy is 75.2%
   Using feature(s) {1, 2, 3, 5, 6, 7, 8, 11, 12} accuracy is 78.4%
   Using feature(s) {1, 2, 3, 5, 6, 7, 8, 9, 12} accuracy is 76.8%
   Using feature(s) {1, 2, 3, 5, 6, 7, 8, 9, 11} accuracy is 75.6%

::::            //I deleted about 2 pages of the trace to save space


   Using feature(s) {5, 7, 8, 12} accuracy is 78.0%
   Using feature(s) {3, 7, 8, 12} accuracy is 72.6%
   Using feature(s) {3, 5, 8, 12} accuracy is 87.6%
   Using feature(s) {3, 5, 7, 12} accuracy is 89.4%
   Using feature(s) {3, 5, 7, 8} accuracy is 89.4%
Feature set {3, 5, 7, 12} was best, accuracy is 89.4%

   Using feature(s) {5, 7, 12} accuracy is 82.8%
   Using feature(s) {3, 7, 12} accuracy is 71.0%
   Using feature(s) {3, 5, 12} accuracy is 91.2%
   Using feature(s) {3, 5, 7} accuracy is 91.8%
Feature set {3, 5, 7} was best, accuracy is 91.8%

   Using feature(s) {5, 7} accuracy is 84.6%
   Using feature(s) {3, 7} accuracy is 73.4%
   Using feature(s) {3, 5} accuracy is 94.4%
Feature set {3, 5} was best, accuracy is 94.4%

   Using feature(s) {5} accuracy is 83.0%
   Using feature(s) {3} accuracy is 74.4%
(WARNING, Accuracy has decreased! Continuing search in case of local
maxia)
Feature set {5} was best, accuracy is 83.0%

Finished search!! The best feature subset is {3, 5}, which has
accuracy of 94.4%
```