Project 2 CS205: Feature Selection with Nearest Neighbor

Github - https://github.com/sshiv012/FeatureSelection

Team Member 1: Bhavya S. Gada | SID: 862385203
Team Member 2: Suryaa Charan Shivakumar | SID: 862395094

Contributions:
Bhavya Gada: Forward Selection, Speed Up Optimizations (Pruning, Sampling), Report Writing
Suryaa Charan Shivakumar: Driver, Nearest Neighbor, Backward Elimination, Report Writing

Datasets:
Small dataset: CS170_small_Data__19.txt (Birthday: 12/**19**/1998)
Large dataset: CS170_large_Data__23.txt (Birthday: 11/**23**/1997)
XXXlarge dataset: CS170_XXXlarge__23.txt (**12**/19/1998 and **11**/23/1997 => **12**+**11**)

For completing this assignment we used the following resources:

- Python 3 documentation: https://docs.python.org/3/
- Lecture slides on Nearest Neighbor classification and speed up feature search
- To read csv files in python https://stackoverflow.com/questions/41585078/how-do-i-read-and-write-csv-files

All important code is original, unimportant subroutines that are not completely original are:

- Subroutines from **math** to calculate the euclidean distance. (sqrt)
- Subroutines from **time** to keep track of how long a particular search runs (time)
- Subroutines from **csv** to read the real-world csv dataset
- Subroutines from **random** to shuffle the dataset before sampling (shuffle)

Table of Contents:

Introduction

Nearest neighbor classification is a simple and effective algorithm that works on the principle of closeness or similarity between the data instance to be classified and the training data instances. A class label is assigned to the instance based on the class labels of its nearest neighbors in the training data. For example, in Figure 1, we need to classify the given insect (purple) as either a grasshopper (blue) or a katydid (red) based on its antenna and abdomen length. As we can see, the new instance is closer/similar to the katydids than the grasshoppers. Therefore, we classify it as a katydid.

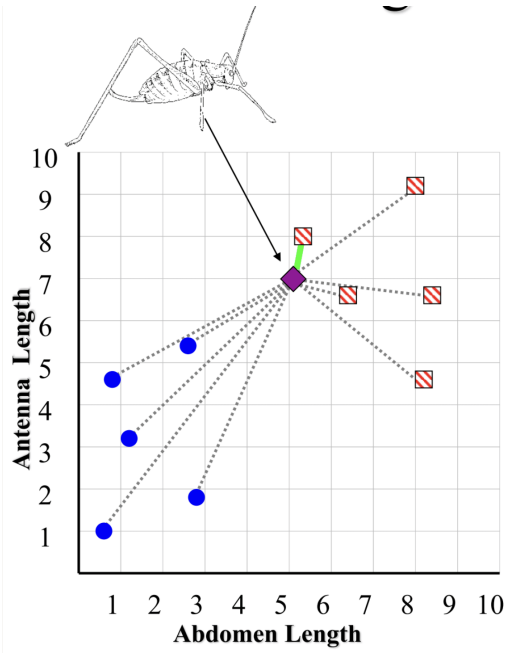

Figure 1. Example of nearest neighbor classification

In our example, we were given two features, but we don't have to use both of them. Using either of the features individually could also work. To pick the best, we could just try all the possible combinations of features required. However, when we have tens or hundreds of features, we need to select the best subset among them. This is where we apply search.

In this report, we first provide a brief overview of the search algorithms used (forward search and backward elimination). Then, we show a comparison of their performance and share some of our observations.

Figure 1: Lecture slides https://rb.gy/7wbhw (slide 66)

Real world dataset

We did a forward search on Breast Cancer Wisconsin (Diagnostic) Data Set, this dataset was taken from https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data. We chose this dataset as it was part of our Data Mining Techniques CS235 Project

Feature Information:

1) ID number
2) Diagnosis (M = malignant, B = benign)
3-32) Ten real-valued features are computed for each cell nucleus:
a) radius (mean of distances from center to points on the perimeter)
b) texture (standard deviation of gray-scale values)
c) perimeter
d) area
e) smoothness (local variation in radius lengths)
f) compactness (perimeter^2 / area - 1.0)
g) concavity (severity of concave portions of the contour)
h) concave points (number of concave portions of the contour)
i) symmetry
j) fractal dimension ("coastline approximation" - 1)
The mean, standard error and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.
All feature values are recorded with four significant digits.
Class distribution: 357 benign, 212 malignant (569 instances)

All feature values are discrete, we changed the classes benign to 0 and malignant to 1.
We remove the ID attribute as it is just a representation of the row indexes.

When we ran a forward search, the algorithm suggested the selection of "concave points_worst," "radius_worst," "compactness_worst," and "smoothness_mean" as the features providing the highest accuracy **(93.1%)** in the classification of breast cancer. This can be justified by understanding the biological relevance in relation to the characteristics of breast cancer tumors. The chosen features capture important aspects related to the shape, size, complexity, and irregularity of breast tumors, which are known to be significant factors in the classification of breast cancer as justified below -

1. Concave points: The feature "concave points_worst" represents the number of concave portions of the contour of the tumor. In breast cancer, tumors often exhibit irregular and asymmetrical shapes, which can be characterized by the presence of concave regions. The number of concave points can be indicative of the tumor's complexity and its potential for malignancy.
2. Radius: The feature "radius_worst" represents the mean of distances from the center to points on the perimeter of the tumor. In breast cancer, the size and extent of the tumor can play a crucial role in its classification. Larger tumor sizes (greater radius) are often associated with more advanced stages of cancer and increased likelihood of malignancy.
3. Compactness: The feature "compactness_worst" represents the ratio of the perimeter squared to the tumor's area minus 1.0. Compactness can provide insights into the tumor's shape and density. In breast cancer classification, tumors with higher compactness values may indicate a more irregular and complex shape, which can be associated with malignancy.
4. Smoothness: The feature "smoothness_mean" represents the local variation in radius lengths of the tumor. In breast cancer, smoothness refers to the consistency of the tumor's surface. Tumors with higher smoothness values may have a more regular and uniform shape, which can be indicative of a benign tumor. Conversely, tumors with lower smoothness values may exhibit irregularities and roughness, which can suggest malignancy.

When we ran backward elimination, the algorithm suggested selecting the features "texture_worst" , "perimeter_worst" with a slightly lower accuracy **91.9%** in the classification of breast cancer. This can be justified by understanding the biological relevance in relation to the characteristics of breast cancer tumors as follows:

1. Texture: The feature "texture_worst" represents the standard deviation of gray-scale values in the tumor region. In breast cancer, variations in texture can be indicative of different tissue structures and composition within the tumor. Texture analysis has been explored in medical imaging to detect subtle changes in tissue patterns, which can aid in identifying abnormal regions
2. Perimeter: The feature "perimeter_worst" represents the total length of the tumor's perimeter. In breast cancer classification, the perimeter provides information about the boundary of the tumor. Tumors with larger perimeters are generally associated with more extensive and irregular shapes, which can be indicative of malignancy.

In our observation, the difference in accuracy between the forward search and backward elimination strategies can be attributed to their inherent characteristics and the specific dataset used in the breast cancer classification.

During the forward search, we noticed that the algorithm begins with an empty set of features and gradually adds the most promising feature at each step. This approach tends to prioritize features that have a significant impact on the classification performance, resulting in the selection of a larger set of features. As a result, the forward search achieved a higher accuracy of 94 percent by including features such as "concave points_worst," "radius_worst," "compactness_worst," and "smoothness_mean."

In contrast, the backward elimination strategy starts with all features and eliminates the least promising feature at each step. This iterative process aims to identify the subset of features that are most essential for accurate classification. However, we observed that during the elimination process, some relevant features were removed, which led to a slightly lower accuracy compared to the forward search. These differences in accuracy can be attributed to the nature of the search strategies and the specific characteristics of the breast cancer dataset. The forward search prioritizes early improvements in accuracy by progressively adding features, while the backward elimination focuses on identifying the essential features by iteratively eliminating them. It is important to note that the selection of features ultimately depends on the dataset and the specific task at hand, and further analysis and validation are required to determine the most optimal feature set for breast cancer classification.

In our observation, the algorithm justified the selection of the "worst" or largest values of certain features, such as radius, texture, perimeter, etc., rather than considering the mean or standard error.

By focusing on the "worst" values, the selection can be justified based on its ability to capture the most significant and extreme measurements associated with the tumor. The algorithm considered the variability and heterogeneity of breast tumors, acknowledging that they can exhibit significant differences in their characteristics. By giving emphasis to the "worst" values, the algorithm aimed to capture the most aggressive and aberrant tumor features that may have a greater impact on the classification accuracy compared to measures of central tendency or dispersion.

Search Algorithms

Forward Search

Forward search involves performing an exhaustive search over every subset of features, but this approach does not scale well. Instead we perform an exhaustive search greedily. In the first round, we test each feature individually and pick the best among them. In the subsequent rounds, we combine the best feature subset from the previous round with each of the remaining features. At each level of search, we keep track of the best feature subset and the highest accuracy that it provides. All other features would be irrelevant.

Backward Elimination

Backward elimination is an alternative strategy for feature search and is the exact opposite of forward search. It works in a similar greedy manner but instead of starting with an empty set to find the best subset of features, we start with the set of all the features and find the best subset by removing one feature at a time. The feature removed is the one which has the least impact on accuracy. At the end this approach would also provide the best feature subset with the highest accuracy. All other features would be irrelevant.

Comparison of Algorithms (After optimizations)

Optimizations performed:

1. 50% sample data used after random shuffle.
2. “minimax like” pruning.
3. Early abandoning if accuracy decreases in two consecutive levels (only for forward search).

	Small			Large			XXXLarge			Real-world		
	Features	Accuracy (%)	Time (s)	Features	Accuracy (%)	Time (s)	Features	Accuracy (%)	Time (s)	Features	Accuracy (%)	Time (s)
Forward Search	**<2, 10>**	**96.6**	**26.28**	**< 16, 19>**	**97.7**	**221.15**	**<65, 20>**	**98.1**	**799.85**	**<28,21,26,5>**	**93.1**	**40.13**
Backward Elimination	**<2, 10>**	**96.6**	**58.75**	**<16, 19>**	**97.7**	**1545.17**	**Not run**	**Not run**	**Not run**	**<22,23>**	**91.9**	**413.88**

Table 1. Comparison of Search Algorithms w.r.t time taken on each dataset on MAC M2, 8GB RAM

Conclusion

In summary, we successfully implemented and evaluated forward search and backward elimination feature search algorithms. Initially, the algorithms faced challenges in processing large datasets. To overcome this, we introduced optimizations including sampling, MinMax-based pruning, and early abandoning, resulting in exponential speedup and improved processing efficiency. While there was a slight variance in accuracy on synthetic datasets, the optimized algorithms demonstrated significantly enhanced processing speed. The concept of early abandoning, driven by the understanding that substantial improvements are often observed during the early stages of the search process, played a pivotal role in achieving an exponential speedup. We also applied the algorithms to the Wisconsin Breast Cancer dataset, obtaining valuable insights into feature relevance for classification. The outcomes of this analysis provided valuable insights into the selected features and their corresponding classification accuracies, thereby contributing to the understanding of feature relevance in breast cancer classification.

Outputs

Example output traceback on the small dataset (Forward search)

```
bhavya@Bhavyas-MacBook-Pro FeatureSelection % python3 FeatureSelection.py
Welcome to Suryaa/Bhavya's Feature Selection program.
Type in the name of the file to test, type default to run on Wisconsin Breast
Cancer Dataset :
CS170_small_Data__19.txt
Type the number of the algorithm you want to run
1) Forward Selection
2) Backward Elimination
1
This dataset has 10 features(not including class attribute), with 1000
instances
Running nearest neighbor with all 10 features, using "leaving-one-out"
evaluation, we get an accuracy of 75.400%
Do you want to stop early if there are consecutive decreases in accuracy
(OPTIMIZATION)? (y/n)
y
Beginning forward search.
Using feature(s) {1} accuracy is 71.600%
Using feature(s) {2} accuracy is 82.800%
Using feature(s) {3} accuracy < 82.800%
Using feature(s) {4} accuracy < 82.800%
Using feature(s) {5} accuracy < 82.800%
Using feature(s) {6} accuracy < 82.800%
Using feature(s) {7} accuracy < 82.800%
Using feature(s) {8} accuracy < 82.800%
Using feature(s) {9} accuracy < 82.800%
Using feature(s) {10} accuracy < 82.800%
Feature set [2] was best, accuracy is 82.800%
Using feature(s) {2, 1} accuracy is 83.800%
Using feature(s) {2, 3} accuracy < 83.800%
Using feature(s) {2, 4} accuracy < 83.800%
Using feature(s) {2, 5} accuracy is 84.100%
Using feature(s) {2, 6} accuracy is 84.200%
Using feature(s) {2, 7} accuracy is 86.100%
Using feature(s) {2, 8} accuracy < 86.100%
Using feature(s) {2, 9} accuracy < 86.100%
Using feature(s) {2, 10} accuracy is 96.600%
Feature set [2, 10] was best, accuracy is 96.600%
Using feature(s) {2, 10, 1} accuracy is 93.400%
Using feature(s) {2, 10, 3} accuracy is 94.500%
Using feature(s) {2, 10, 4} accuracy < 94.500%
Using feature(s) {2, 10, 5} accuracy < 94.500%
```

```
Using feature(s) {2, 10, 6} accuracy < 94.500%
Using feature(s) {2, 10, 7} accuracy < 94.500%
Using feature(s) {2, 10, 8} accuracy < 94.500%
Using feature(s) {2, 10, 9} accuracy < 94.500%
(Warning, Accuracy has decreased! Continuing search in case of local maxima)
Feature set [2, 10, 3] was best, accuracy is 94.500%
Using feature(s) {2, 10, 3, 1} accuracy is 90.000%
Using feature(s) {2, 10, 3, 4} accuracy is 90.200%
Using feature(s) {2, 10, 3, 5} accuracy is 90.400%
Using feature(s) {2, 10, 3, 6} accuracy is 90.700%
Using feature(s) {2, 10, 3, 7} accuracy < 90.700%
Using feature(s) {2, 10, 3, 8} accuracy < 90.700%
Using feature(s) {2, 10, 3, 9} accuracy < 90.700%
(Warning, Accuracy has decreased! Continuing search in case of local maxima)
Feature set [2, 10, 3, 6] was best, accuracy is 90.700%
Accuracy has decreased for two consecutive steps. Stopping early.
Finished search!! The best feature subset is [2, 10], which has an accuracy of
96.600%
Finished forward search in 26.28 seconds.
```

Output on the large dataset (Forward search)

```
Accuracy has decreased for two consecutive steps. Stopping early.
Finished search!! The best feature subset is [16, 19], which has an accuracy of 97.700%
Finished forward search in 221.15 seconds.
```

Output on the XXXlarge dataset (Forward search)

```
Accuracy has decreased for two consecutive steps. Stopping early.
Finished search!! The best feature subset is [65, 20], which has an accuracy of 98.100%
Finished forward search in 799.85 seconds.
```

Output on the real-world dataset (Forward search)

```
Accuracy has decreased for two consecutive steps. Stopping early.
Finished search!! The best feature subset is [28, 21, 26, 5], which has an accuracy of 93.100%
Finished forward search in 40.13 seconds.
bhavya@Bhavyas-MacBook-Pro FeatureSelection % python3 FeatureSelection.py
```

Output on the real-world dataset (Backward elimination)

```
Finished search!! The best feature subset is {22, 23}, which has an accuracy of 91.900%
Finished backward elimination search in 413.88 seconds.
```

*Console outputs have been intentionally omitted for some backward eliminations to save up space. Also the code has been omitted, but it has been commented and documented in detail in the github repository provided on the first page (under the project title)