Project 2: Feature Selection with K Nearest Neighbors

Contributors

Jon Darius, Rohan Gujral, Jerry Li, Vaneesha Singh, Aditi Thanekar

External Resources

https://www.geeksforgeeks.org/k-nearest-neighbours/
https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/
https://medium.com/analytics-vidhya/mean-normalization-and-feature-scaling-a-simple-explanation-3b9be7bfd3e8
https://towardsdatascience.com/backward-elimination-for-feature-selection-in-machine-learning-c6a3a8f8cef4

Imports

math, numpy, heapq, copy, random, matplotlib, seaborn, sklearn.preprocessing (strictly for post-hoc analysis using PCA), MinMax Scaling from sklearn, StandardScaling from sklearn

Different Contributions

Aditi/Vaneesha

Backward Elimination feature selection algorithm and design and implementation of the actual NN classifier, bidirectional selection

Rohan/Jerry

Greedy Forward feature selection algorithm as well as design and implementation of the validator class with leave one out validator and displaying accuracies, bidirectional selection, making plots

Jon

Created visualizations of best/worst performing features for all datasets and feature search algorithms. Performed both prior analysis of datasets and post-hoc analysis on the performance of NN. Main contribution to the report.

I. Introduction

For this project, we will develop a Nearest Neighbor (1NN) machine learning model to classify data points labeled as 1.0 or 2.0. The NN classifier is a popular supervised learning algorithm known for its simplicity and effectiveness. It uses distance metrics and the nearest instance to classify new instances based on the labels of their nearest neighbors.

In our implementation, we will use the Euclidean distance metric to measure the distances between new instances and their nearest neighbor. Additionally, we will perform thorough data analysis both before and after training to evaluate the performance of our model. This analysis will help us determine the effectiveness of NN on both small and large datasets, ensuring we identify the best-performing models for various data sizes.

We will also utilize various validation techniques, like leave-one-out cross-validation, to identify both the highest-performing model and the simplest model to avoid overfitting. We also implemented K-fold cross-validation which works by splitting the training data into K subsets, or folds in the beginning but we didn't need it in the end because we changed it to leave one out. In each iteration, one fold is used as the validation set while the remaining K-1 folds are used for training. This process is repeated K times, with each fold used exactly once as the validation set, ensuring a comprehensive evaluation of the model's performance. In addition, we have leave-one-out cross validation which is a special case of K-fold where K is equal to the number of data points, meaning each iteration uses a single data point as the validation set and the rest as the training set, providing another optimal evaluation of the model.

Additionally, to identify the most optimal set of features, we will employ two feature selection algorithms: forward selection and backward elimination. Forward selection starts with an empty set of features and iteratively adds the feature that results in the highest increase in model accuracy, continuing this process until the most optimal subset of features is selected. Backward elimination, on the other hand, begins with the full set of features and iteratively removes the feature that contributes the least to model accuracy. Both techniques aim to achieve optimal model performance by eliminating ineffective features and retaining only those that significantly contribute to predictive accuracy.

II. Challenges

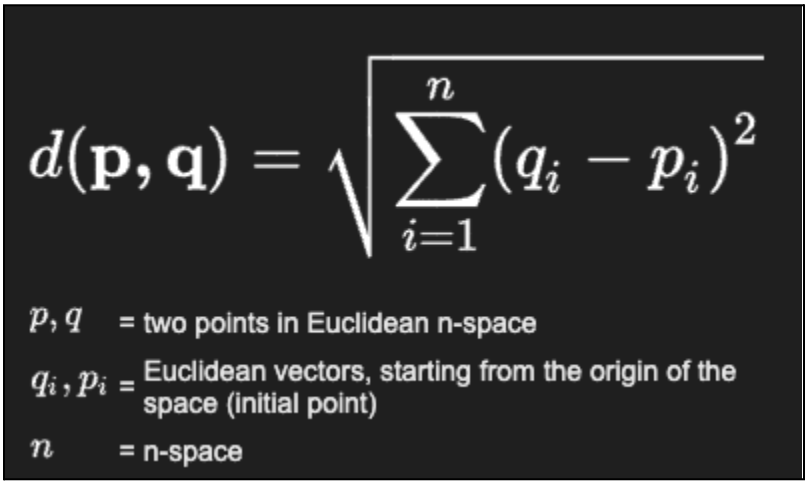
One of our first challenges with this project was to make our classes compatible with each other, since there was some miscommunication during the design process between some of the group members. It took a while to fully understand why the classes were not compatible with each other and it boiled down to inconsistencies within their individual designs. For example, our Problem class was not able to properly access the Node data members and while testing our validator, we were unable to train our classifier because of how the data was supposed to be read

in and how it was actually read in. Another issue that we had was generating visuals that would add to the overall predictive accuracy of our model. For example, we didn't know which normalization technique to select for visualizing our data between different features; essentially wondering which measure would help us paint the best picture.

III. Code Design

For our project we ended up using **python** to implement our functions. The reason why we chose python was because we felt that the ease of visualization in python was far better than that of C++ or any other programming language. Furthermore, here is a breakdown of our code design:

1. Main Function
    a. This function acts as an entry point for the user to input the amount of features they want to train on.
    b. It loads in a dataset from file path and creates a Problem object (will cover in a second) alongside a classifier object.
    c. The user is then asked to select either forward selection or backward elimination to use as their feature search algorithm.
    d. We then output the trace of each feature selection, the accuracy for each step, and the overall time it takes to calculate predictive accuracy.
2. Classifier.py (Classifier Class)
    a. This class contains both our euclidean distance function and our nearest neighbor algorithm
    b. For our euclidean distance algorithm, we take in 2 data points as arguments and we check to see what type of object instances they are to ensure we don't run into any calculation errors when calculating the distance between the 2 data point types. We then return the euclidean distance using the following formula:



    c. Our nearest neighbor algorithm takes in the data that we want to train on, the new instance, and the feature subset returned by our search algorithm as arguments. For each data point we calculate the distance between each data point and the

new instance. During this process, we also see if the newly calculated distance is lower than the minimum distance, and if it is we assign the nearest instance to that new instance. We then return the class label of the nearest neighbor.

   d. We also implemented NN as an optimization algorithm to the nearest neighbor, which takes the majority of the K class classifications.

3. Searches.py (Problem Class)
   a. This class consists of a “search problem” where we take in a list of features to perform a search algorithm on. The functions in this class are the eval function which returns the accuracy of our leave one out validator. In this class we also have both the forward selection and backward elimination algorithms.
4. Validation.py (Validator Class):
   a. Our leave-one-out function implements leave-one-out cross-validation and takes the classifier and the feature set as arguments. This validation technique uses all data points except one. This process is repeated for each data point, and the accuracy of the model is calculated as the proportion of correct predictions. The function returns the mean accuracy across all iterations.
   b. Our k_fold function implements K-fold cross-validation with the classifier, the feature set, and K as arguments. This function uses a technique where the data is divided into K subsets, or folds. The model is then trained on K-1 folds and validated on the remaining fold, with this process repeated K times. The function first applies min-max normalization to each feature in the dataset. It then divides the data into K folds, trains the classifier on K-1 folds, and validates on the remaining fold. The accuracy of the model is calculated for each fold, and the average accuracy is printed for each iteration of the K-fold process.
5. Visualize Features (Script File):
   a. This file doesn’t contain any class implementations. A big part of this file is converting each of the text files into panda dataframes for ease of access to creating visualizations of the data.
   b. There are functions like visualize_1d, visualize_2d, high_dimensional which are used to visualize the best and the worst features for predictive accuracy. These functions allow us to conduct pre-analysis of our data and also ad-hoc analysis that can help us visualize the relationships between selected features.

      For the high_dimensional function, we utilize PCA analysis to reduce the data's high dimensional space to a lower-dimensional space to better visualize the trends in data that help support our calculated accuracies that use large feature subsets.

IV. Dataset details

When it came to our datasets, we had a float classification type that consisted of only 1.0 or 2.0. In terms of the type of data we had, all of our data was floating point numbers. This applies to all of the following datasets.

The General Small Dataset

Number of Features	Number of Instances in Dataset
10	100

The General Large Dataset

Number of Features	Number of Instances in Dataset
40	1000

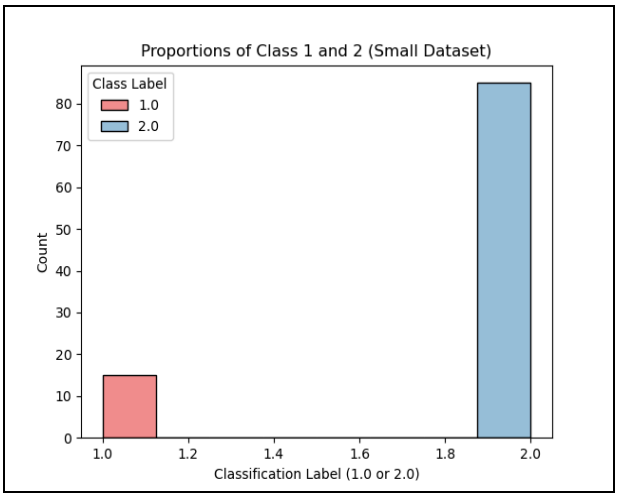
Our Personal Small Dataset

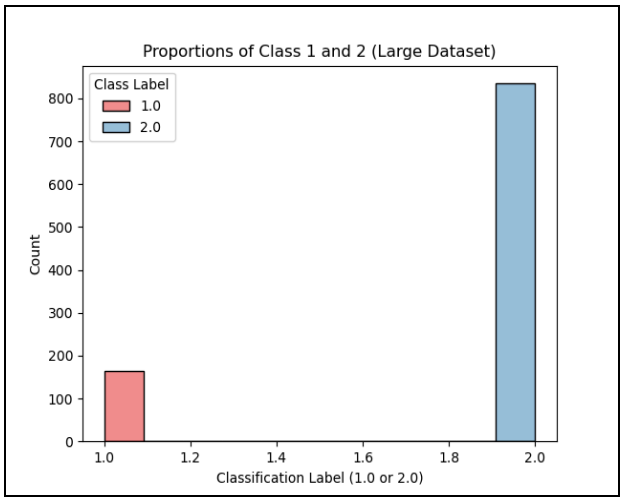
Number of Features	Number of Instances in Dataset
10	100

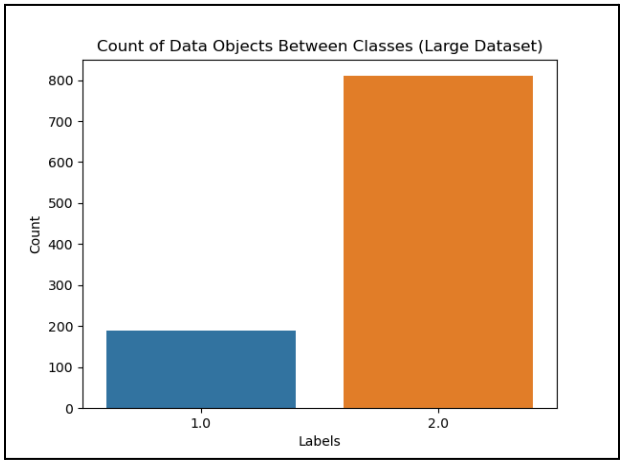
Our Personal Large Dataset

Number of Features	Number of Instances in Dataset
40	1000

Visuals Showing Class Imbalance

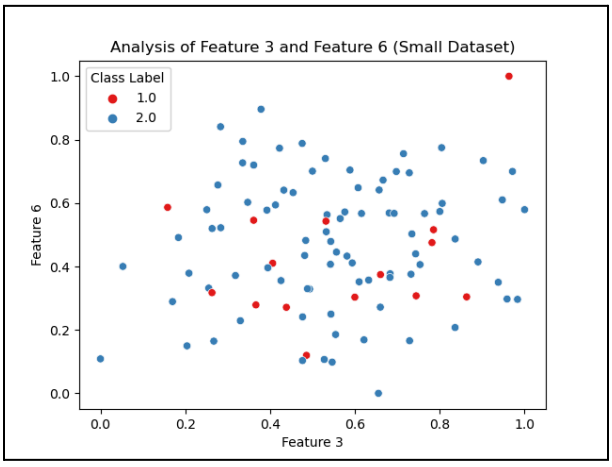


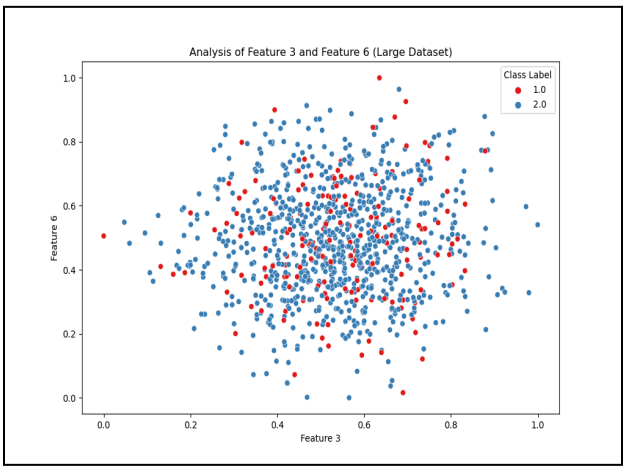




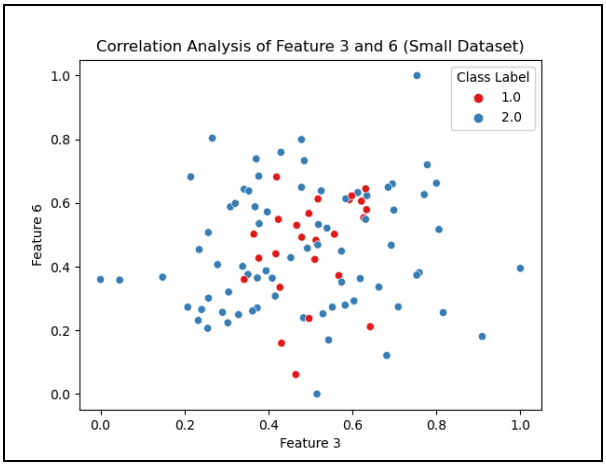
For the prior analysis of our datasets, I wanted to see the proportion of what labels we had for our data. When looking at the proportions of labels for the small dataset, we can see that there are far more instances of 2.0 than 1.0, which could possibly lead to bias in classifying new instances since the majority of the labels are 2.0. This point is furthered when looking at the large dataset graph, where we can see the same trend. The graphs on the bottom also represent this same trend for the non-custom datasets. Overall, there is massive class imbalance which could cause bias in our NN model.

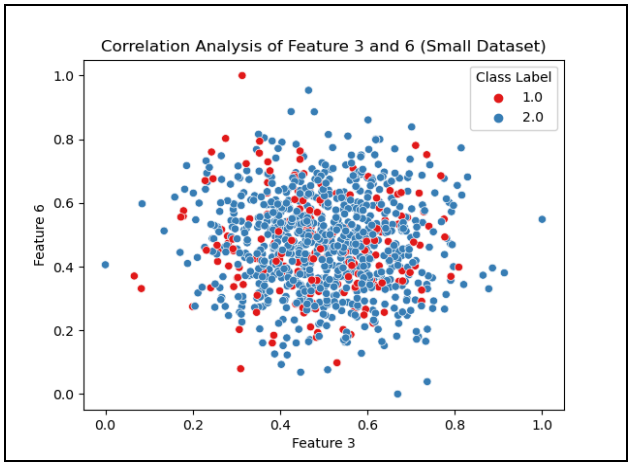
Visuals Showing Non-Correlated Features (No Trends)





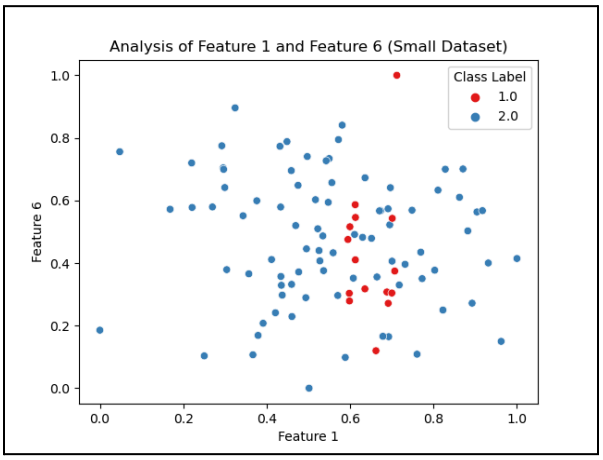
Non Custom Datasets

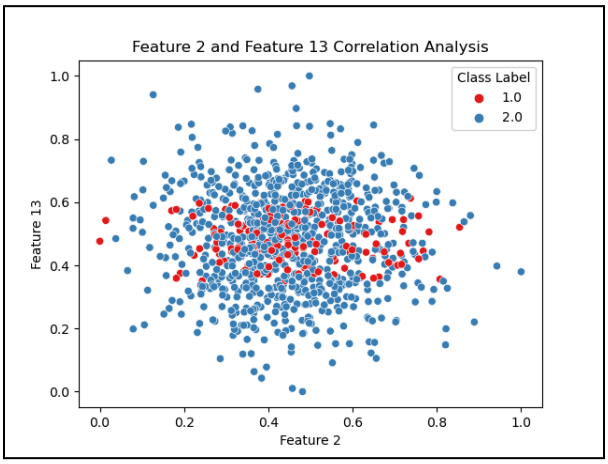




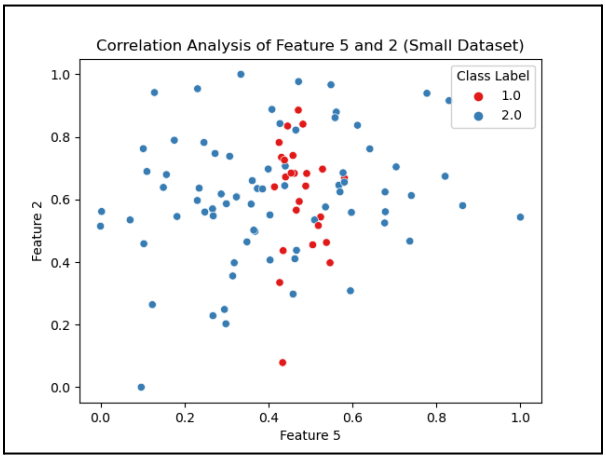
When analyzing our data before classification we can see that certain features show no trends or any signs of correlation. For example, for features 3 and 6 we can see that there are no trends amongst the data points. This trend applies for both the small and large datasets since you can see that the 1.0 labels and 2.0 labels are equally as scattered. The same thing applies for the 2 graphs on the bottom which represent features 3 and 6 in the non-custom datasets. Both graphs on the bottom do not show any correlation between features, such that we can not reach a conclusion on their trends.

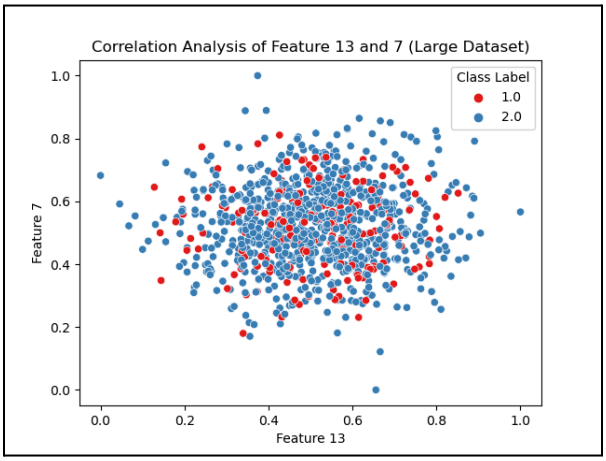
Visuals Showing Correlated Features (Distinguishable Trends)





Non Custom Datasets





When analyzing trends among different features, we observed a distinct pattern between Feature 1 and Feature 6 in the small dataset. Specifically, Feature 1's 1.0 labels tend to cluster between 0.6 and 0.8, indicating a clear trend, whereas Feature 6 shows no present trend. In the large dataset, examining Features 2 and 13 reveals a notable trend in the red labels. The red labels consistently fall between 0.3 and 0.6, suggesting a sideways trend within this range, while the blue dots do not exhibit a definitive trend and are more scattered. These observations highlight the varying degrees of trend visibility across different features in both the small and large datasets, providing valuable insights into the data's underlying structure.

V. Algorithms

1. Forward Selection

Start by individually calculating the accuracies of each feature. From there, add features one by one while keeping track of the most promising nodes with a max heap. Expand the most promising nodes from the top of the max heap. We will keep using the greedy approach to add the most helpful feature until a certain predefined evaluation condition is met.

2. Backward Elimination

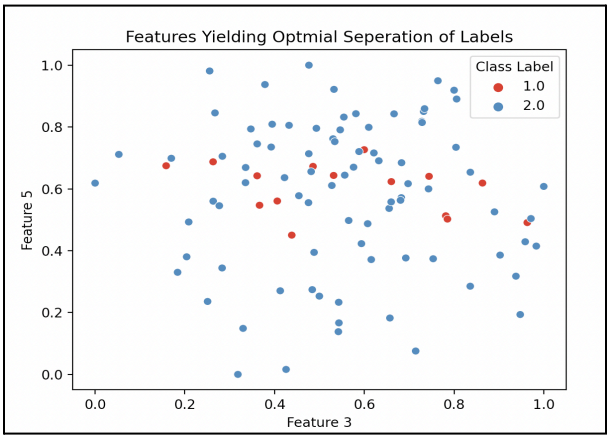
Start with all of the features, and calculate the accuracy of using all the features. Then you can branch out by removing one feature at each child node and recalculate the accuracies with the new feature subsets. If the accuracy goes down in comparison with the parent, then stop searching. Otherwise continue removing one different feature from the parent’s feature subset for each child node and calculate the accuracy.

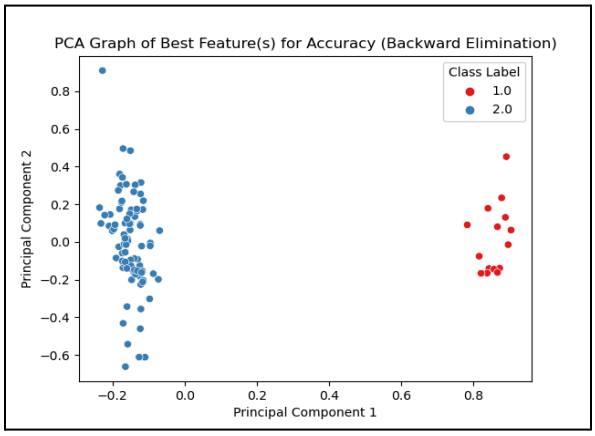
3. Bidirectional Selection

Bidirectional selection is a combination of both Forward Selection and Backward Elimination. It starts out with no features and then iteratively adds features that improves the models performance and removes any remaining features that do not. It also uses two different threshold parameters that are used to determine if a new feature should be added based on their improvement to the previous one. By combining the properties of adding and removing features it gives a good balance between performance and the runtime. In our tests, Bidirectional Selection would have an edge over Forward Selection most of the time in terms of runtime, while having the same performance.

VI. Analysis

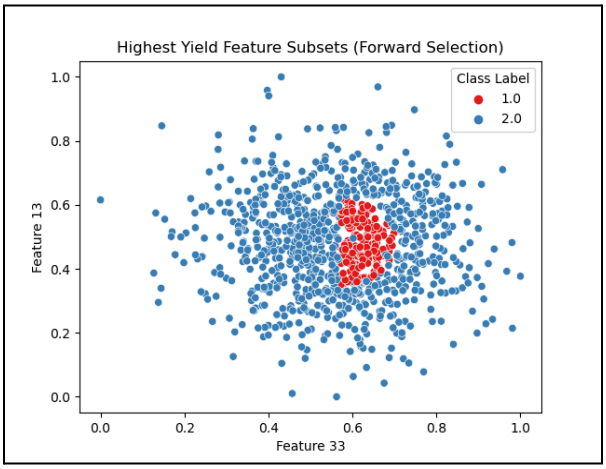
Experiment 1a: Comparing Forward Selection vs Backward Elimination (Small Dataset)

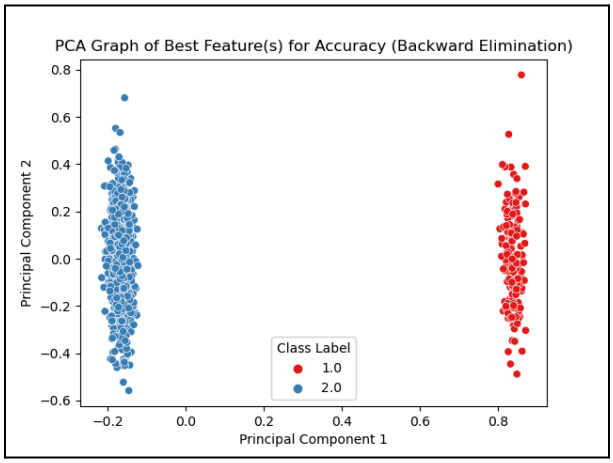




When analyzing the difference between forward selection and backwards elimination, it becomes more evident that backward elimination yields the better set of results. For our forward selection, it selected features 3 and 5 as the best subset of features for the NN model, where the red label creates a linear trend between values .5 and .7 on the y-axis. The red points seem sparse, but when encountered closely with distance, it is easy to classify on the nearest instances given some new instance. On the other hand, we see backward elemination picking feature subset [1, 2, 3, 4, 5, 6, 7, 8, 10] which forces us to move the data from a higher dimensional space to a lower dimensional space. For this we ended up using PCA, and the results show that this subset of features create a solid separation between both class labels, meaning that these features allow for simple and accurate instance prediction.

Experiment 1b: Comparing Forward Selection vs Backward Elimination (Large Dataset)





For our large dataset, we can see that the accuracy yielded by both forward selection and backwards selection is optimal. For forward selection, we can see that the red labels tend to cluster together towards the center while the blue dots tend to cluster around the red ones. The highest yielding features for predictive accuracy for forward selection were features 33 and 13. When analyzing the results from the backward elimination, we used all features except feature 33 and the graph on the right yields the following results. Since we were dealing with extremely high dimensionality, we reduced the points to a lower-dimensional space which allows us to see the separation of labels. We can see how well the labels are separated when using the following labels.

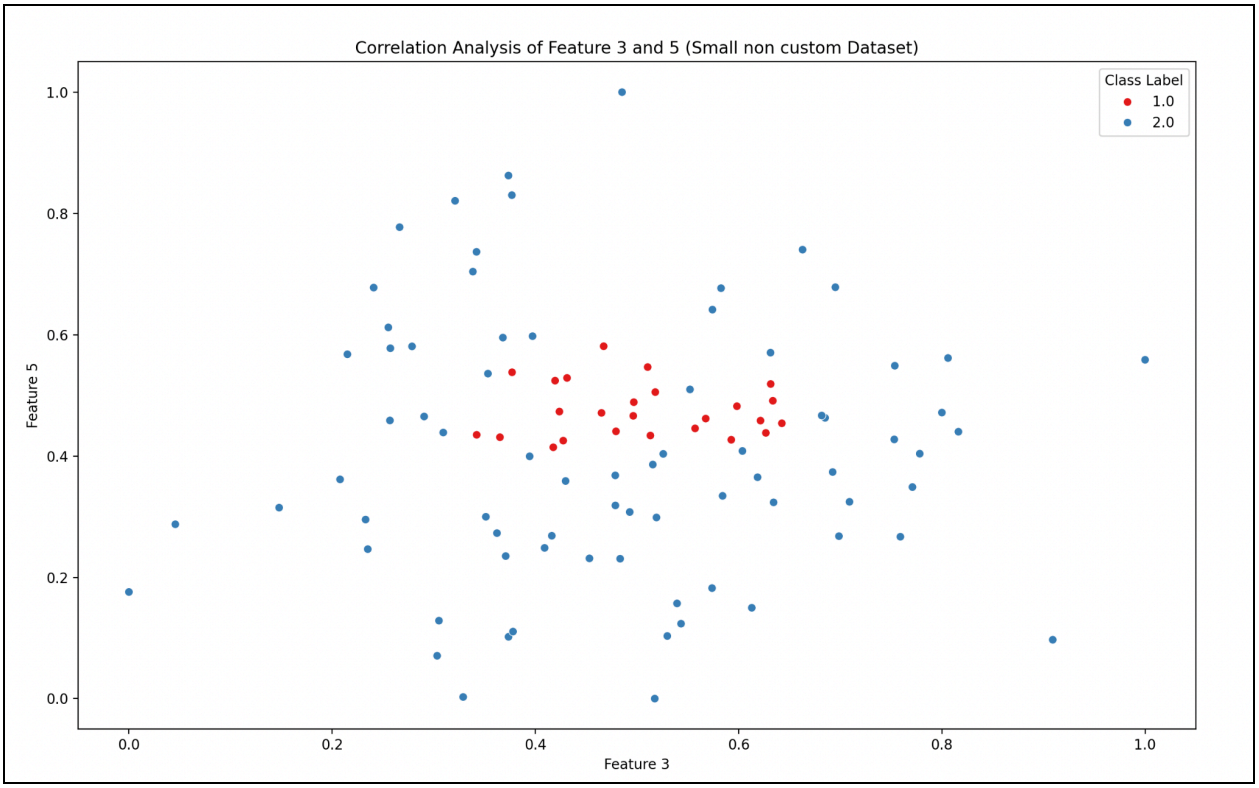
Experiment 1c: Accuracies w/o Feature Selection (Small Dataset & Large Dataset)

```
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  accuracy is  0.68
```

```
Using feature(s) [1, 2, 3, 4, 5, 6, 7, 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] accuracy is 0.69
```

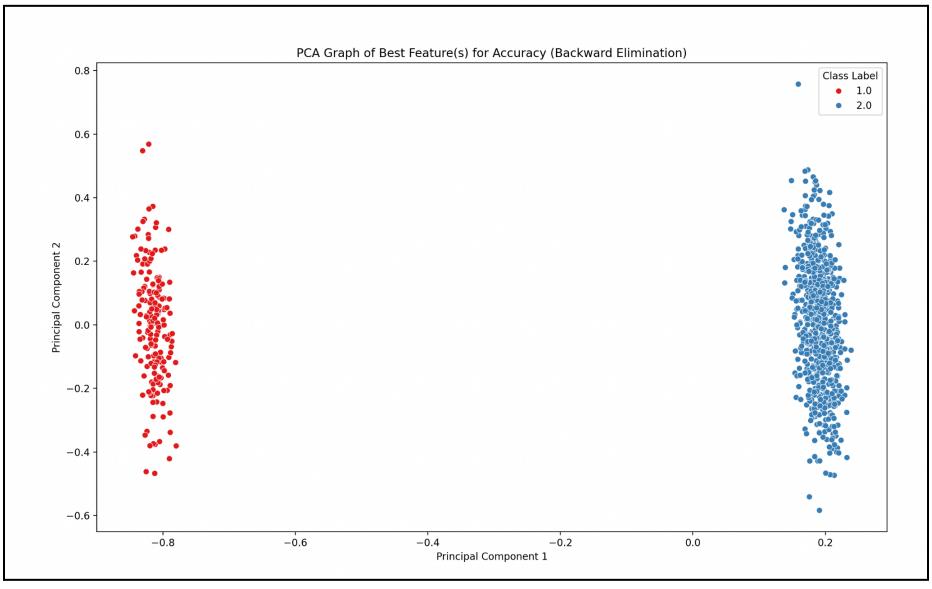
When analyzing the accuracies between feature selection and no feature selection, we see that the classifications with feature selection often perform far better than those without. When looking at the accuracies for no feature selection for both the small and large datasets varies around 68% - 69%, which isn’t optimal. When applying feature selection, we see accuracies of about 86% to even as high as 98%, which improves as we add features. Ultimately, feature selection is an efficient way of removing irrelevant features that don't contribute to model performance.

Experiment 1d: Comparing Best Features of Forward Selection (Small Non-custom Dataset)



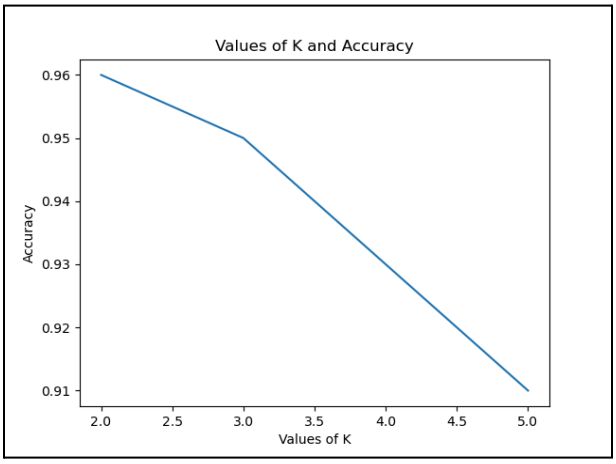
From the above graph, we can see a clear distinction between the two classes through the clustering of red dots in the center, surrounded almost entirely by blue dots. Using features three the red values range primarily from 0.3 to 0.6, and its feature 5 values range mostly from 0.4 to 0.6. The blue values have much greater variation between data points and don't necessarily have defined upper and lower bounds for either of the features because they range from 0 to 1 for features 3 and 5.

Experiment 1e: Comparing Best Features of Forward Selection (Large Non-custom Dataset)



The above graphs show a comparison of features 3 and 5 for the small global dataset and the PCA dimensionality reduction for the global large dataset respectively. By looking at the unique and interesting clustering of the data, we can see clear and definitive boundaries between each of the two classes. You can see the distinction where class 1 values are correlated to the left with a PC1 value less than -0.6 and with the values of class 2 being correlated to the right above PC1 value of 0. After reducing to the two most relevant features, we can still observe a boundary in the form of a circle simply using 2 out of the 10 features. From the second graph, we can see the very distinct clusters when using all of the 40 features except for feature 27. This

Experiment 2: Comparing Accuracies to Different Values of K (Small Dataset/Forward Selection)

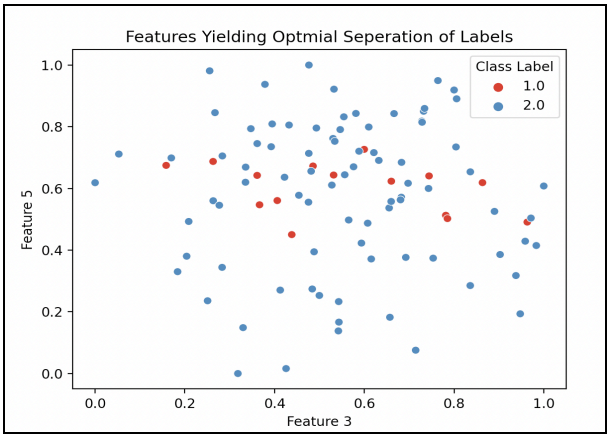


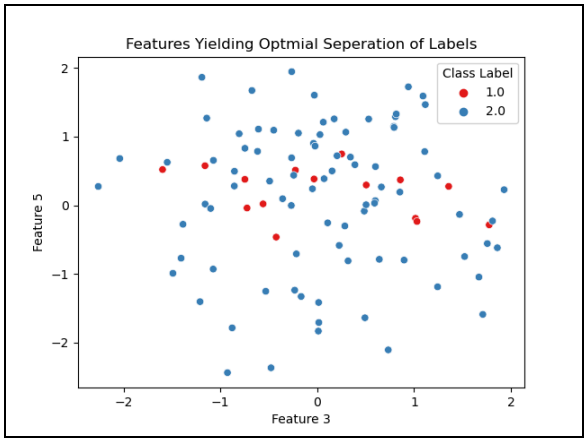
When testing our KNN models with different values of K, we analyzed each accuracy that occurs from each K. For this, we only used the small dataset (for time's sake) and the Forward Selection. The reason for this is because we know that as K increases after some optimal K, it will decrease the performance of our KNN algorithm. So regardless of which feature selection and dataset we use, it will yield the same trend (maybe not with the exact accuracies). Overall, we do see that after a K value of 2 the accuracy is starting to decrease since it starts both becoming biased towards the dominating class and also starts becoming too sensitive to noise (i.e. overfitting).

Experiment 3: Comparing Both Feature Search Algorithms

When comparing both forward selection and backward elimination, we can see both some evident differences and similarities. Both algorithms operate based on heuristics, meaning they do their operations based on a features evaluation accuracy. In terms of better performance, the forward search seems to perform better with 92% accuracy on the small dataset and then 96% on the large dataset. Backward elimination on the other hand got an accuracy of 75% on the small dataset and 85% on the large dataset. The reason why I think the backward elimination didn't work as well as the forward selection is because its feature subset is far larger than the forward selection, which probably leads to some irrelevant features still being present. Overall, the forward selection ended up performing the best out of the 2 search algorithms we used.

Experiment 4: Comparing Both Data with and without Normalization





When analyzing both of these graphs, we can see the evident difference between data that is normalized and not normalized. The type of normalization we used was min max normalization, and you can see that the min max normalization stretches our data to better see evident trends in labeled data. This might improve how we both analyze our data and how the NN model is able to detect nearest instances between distances.

VII. Conclusion

In conclusion, our project demonstrates that NN is an efficient algorithm for classifying new instances. To enhance our model's performance, we used search algorithms to eliminate irrelevant features, significantly improving accuracy and providing insights into the effectiveness of forward selection and backward elimination. Considering we were able to achieve an accuracy of up to 90±5% We also incorporated K-fold and leave-one-out validation techniques to select the best features and models for predictive accuracy. Forward selection successfully removed many irrelevant features, while backward elimination was less effective. Additionally, our analysis identified features that reveal underlying trends and those that contribute little to classification accuracy. We noted a significant class imbalance between labels 1.0 and 2.0, which can bias the model. Despite these challenges, our thorough optimization and feature analysis have deepened our understanding of the data and strengthened the overall performance of our NN model. Overall, we are very satisfied with our findings and believe this work lays a solid foundation for further improvements and applications in similar classification tasks. Here is a table of our results:

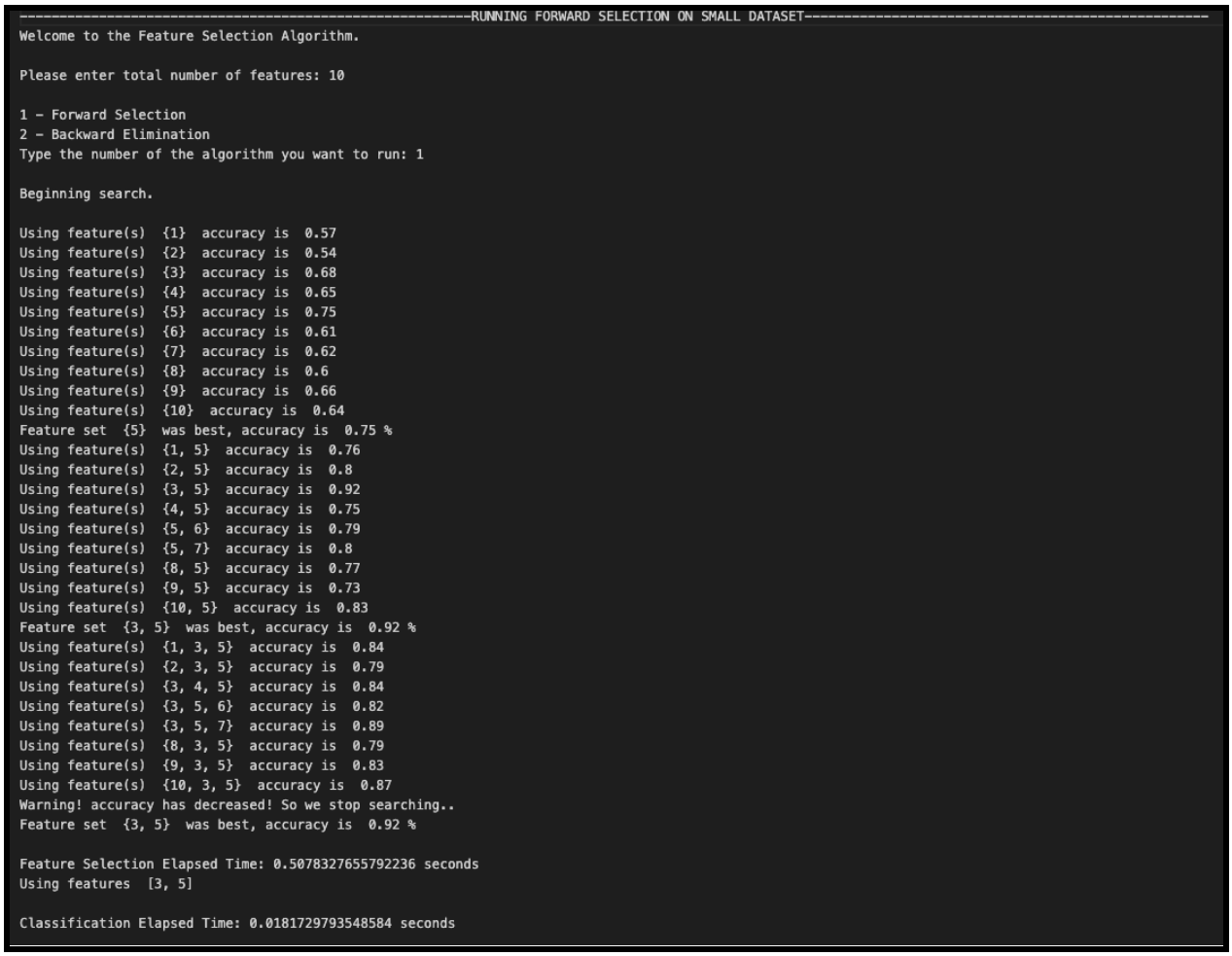
Dataset	Feature Set	Accuracy
Small Dataset: 100 data points	***Forward Selection:*** Features = [3, 5] ***Backward Elimination:***	***Forward Selection:*** 92% ***Backward Elimination:***

Feature #: 10	Features = [1, 2, 3, 4, 6, 7, 8, 9, 10] ***Bidirectional Selection:*** Features = [3, 5]	75% **Bidirectional Selection:** 92%
Large Dataset: 1000 data point **Feature #:** 40	***Forward Selection:*** Features = [1, 27] ***Backward Elimination:*** Features = All Features except 27 ***Bidirectional Selection:*** Features = [1, 27]	***Forward Selection:*** 96% ***Backward Elimination:*** 85% **Bidirectional Selection:** 96%

Dataset	Feature Set	Accuracy
Small Personal Dataset: 100 data points **Feature #:** 10	***Forward Selection:*** Features = [9] ***Backward Elimination:*** Features = [1, 2, 3, 4, 5, 6, 7, 8, 10] ***Bidirectional Selection:*** Features = [9]	***Forward Selection:*** 86% ***Backward Elimination:*** 86% **Bidirectional Selection:** 86%
Large Personal Dataset: 1000 data point **Feature #:** 40	***Forward Selection:*** Features = [33, 13] ***Backward Elimination:*** Features = All Features except 33 ***Bidirectional Selection:*** Features = [33, 13]	***Forward Selection:*** 98% ***Backward Elimination:*** 86% **Bidirectional Selection:** 98%

VIII. Traces of Both Small and Large Dataset

Trace of Small Dataset Using Forward Feature Selection



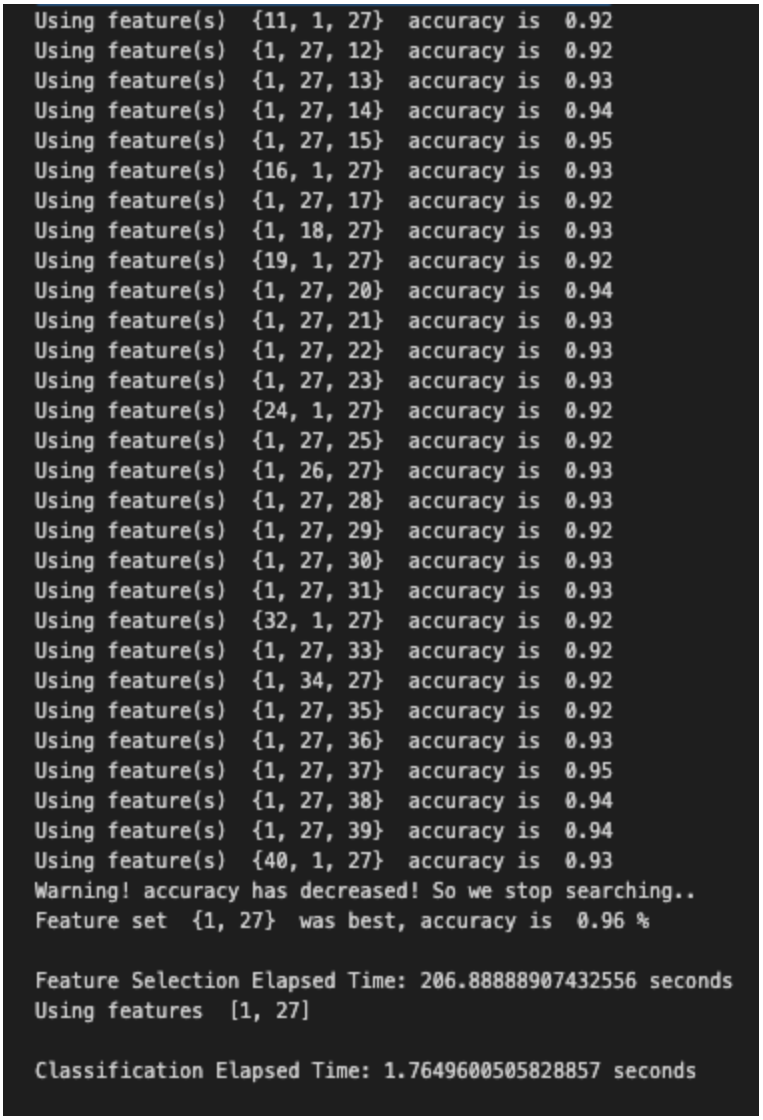
Trace of Small Dataset Using Backward Elimination Feature Selection

```
------------------------------------------------------------RUNNING BACKWARD SELECTION ON SMALL DATASET------------------------------------------------------------
Welcome to the Feature Selection Algorithm.

Please enter total number of features: 10

1 - Forward Selection
2 - Backward Elimination
Type the number of the algorithm you want to run: 2

Beginning search.

Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  accuracy is  0.68
Using feature(s)  [2, 3, 4, 5, 6, 7, 8, 9, 10]  accuracy is  0.57
Using feature(s)  [1, 3, 4, 5, 6, 7, 8, 9, 10]  accuracy is  0.54
Using feature(s)  [1, 2, 4, 5, 6, 7, 8, 9, 10]  accuracy is  0.68
Using feature(s)  [1, 2, 3, 5, 6, 7, 8, 9, 10]  accuracy is  0.65
Using feature(s)  [1, 2, 3, 4, 6, 7, 8, 9, 10]  accuracy is  0.75
Using feature(s)  [1, 2, 3, 4, 5, 7, 8, 9, 10]  accuracy is  0.61
Using feature(s)  [1, 2, 3, 4, 5, 6, 8, 9, 10]  accuracy is  0.62
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 9, 10]  accuracy is  0.6
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 10]  accuracy is  0.66
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9]  accuracy is  0.64
Feature set  [1, 2, 3, 4, 6, 7, 8, 9, 10]  was best, accuracy is  0.75 %
Using feature(s)  [2, 3, 4, 6, 7, 8, 9, 10]  accuracy is  0.57
Using feature(s)  [1, 3, 4, 6, 7, 8, 9, 10]  accuracy is  0.54
Using feature(s)  [1, 2, 4, 6, 7, 8, 9, 10]  accuracy is  0.68
Using feature(s)  [1, 2, 3, 6, 7, 8, 9, 10]  accuracy is  0.65
Using feature(s)  [1, 2, 3, 4, 7, 8, 9, 10]  accuracy is  0.61
Using feature(s)  [1, 2, 3, 4, 6, 8, 9, 10]  accuracy is  0.62
Using feature(s)  [1, 2, 3, 4, 6, 7, 9, 10]  accuracy is  0.6
Using feature(s)  [1, 2, 3, 4, 6, 7, 8, 10]  accuracy is  0.66
Using feature(s)  [1, 2, 3, 4, 6, 7, 8, 9]  accuracy is  0.64
Warning! accuracy has decreased!So we stop searching..
Feature set  [1, 2, 3, 4, 6, 7, 8, 9, 10]  was best, accuracy is  0.75 %

Feature Selection Elapsed Time: 0.22041797637939453 seconds
Using features  [1, 2, 3, 4, 6, 7, 8, 9, 10]

Classification Elapsed Time: 0.03211498260498047 seconds
```

Trace of Large Dataset Using Forward Feature Selection

```
------------------------------------------------------------RUNNING FORWARD SELECTION ON LARGE DATASET------------------------------------------------------------
Welcome to the Feature Selection Algorithm.

Please enter total number of features: 40

1 - Forward Selection
2 - Backward Elimination
Type the number of the algorithm you want to run: 1

Beginning search.

Using feature(s)  {1}  accuracy is  0.74
Using feature(s)  {2}  accuracy is  0.71
Using feature(s)  {3}  accuracy is  0.67
Using feature(s)  {4}  accuracy is  0.68
Using feature(s)  {5}  accuracy is  0.69
Using feature(s)  {6}  accuracy is  0.7
Using feature(s)  {7}  accuracy is  0.71
Using feature(s)  {8}  accuracy is  0.71
Using feature(s)  {9}  accuracy is  0.68
Using feature(s)  {10}  accuracy is  0.69
Using feature(s)  {11}  accuracy is  0.7
Using feature(s)  {12}  accuracy is  0.68
Using feature(s)  {13}  accuracy is  0.66
Using feature(s)  {14}  accuracy is  0.71
Using feature(s)  {15}  accuracy is  0.7
Using feature(s)  {16}  accuracy is  0.68
Using feature(s)  {17}  accuracy is  0.67
Using feature(s)  {18}  accuracy is  0.7
Using feature(s)  {19}  accuracy is  0.7
Using feature(s)  {20}  accuracy is  0.68
Using feature(s)  {21}  accuracy is  0.68
Using feature(s)  {22}  accuracy is  0.68
Using feature(s)  {23}  accuracy is  0.7
Using feature(s)  {24}  accuracy is  0.67
Using feature(s)  {25}  accuracy is  0.69
Using feature(s)  {26}  accuracy is  0.68
Using feature(s)  {27}  accuracy is  0.85
Using feature(s)  {28}  accuracy is  0.66
```

```
Using feature(s)  {29}  accuracy is  0.7
Using feature(s)  {30}  accuracy is  0.71
Using feature(s)  {31}  accuracy is  0.7
Using feature(s)  {32}  accuracy is  0.7
Using feature(s)  {33}  accuracy is  0.7
Using feature(s)  {34}  accuracy is  0.68
Using feature(s)  {35}  accuracy is  0.7
Using feature(s)  {36}  accuracy is  0.7
Using feature(s)  {37}  accuracy is  0.7
Using feature(s)  {38}  accuracy is  0.7
Using feature(s)  {39}  accuracy is  0.71
Using feature(s)  {40}  accuracy is  0.7
Feature set  {27}  was best, accuracy is  0.85 %
Using feature(s)  {1, 27}  accuracy is  0.96
Using feature(s)  {2, 27}  accuracy is  0.84
Using feature(s)  {3, 27}  accuracy is  0.82
Using feature(s)  {27, 4}  accuracy is  0.82
Using feature(s)  {27, 5}  accuracy is  0.82
Using feature(s)  {27, 6}  accuracy is  0.84
Using feature(s)  {27, 7}  accuracy is  0.82
Using feature(s)  {8, 27}  accuracy is  0.85
Using feature(s)  {9, 27}  accuracy is  0.84
Using feature(s)  {10, 27}  accuracy is  0.82
Using feature(s)  {11, 27}  accuracy is  0.82
Using feature(s)  {27, 12}  accuracy is  0.83
Using feature(s)  {27, 13}  accuracy is  0.82
Using feature(s)  {27, 14}  accuracy is  0.83
Using feature(s)  {27, 15}  accuracy is  0.84
Using feature(s)  {16, 27}  accuracy is  0.83
Using feature(s)  {17, 27}  accuracy is  0.84
Using feature(s)  {18, 27}  accuracy is  0.83
Using feature(s)  {19, 27}  accuracy is  0.83
Using feature(s)  {27, 20}  accuracy is  0.82
Using feature(s)  {27, 21}  accuracy is  0.83
Using feature(s)  {27, 22}  accuracy is  0.84
Using feature(s)  {27, 23}  accuracy is  0.84
Using feature(s)  {24, 27}  accuracy is  0.81
Using feature(s)  {25, 27}  accuracy is  0.81
Using feature(s)  {26, 27}  accuracy is  0.83
Using feature(s)  {27, 28}  accuracy is  0.82
Using feature(s)  {27, 29}  accuracy is  0.83
Using feature(s)  {27, 30}  accuracy is  0.82
Using feature(s)  {27, 31}  accuracy is  0.82
Using feature(s)  {32, 27}  accuracy is  0.83
Using feature(s)  {33, 27}  accuracy is  0.82
Using feature(s)  {34, 27}  accuracy is  0.82
Using feature(s)  {35, 27}  accuracy is  0.82
Using feature(s)  {27, 36}  accuracy is  0.82
Using feature(s)  {27, 37}  accuracy is  0.85
Using feature(s)  {27, 38}  accuracy is  0.82
Using feature(s)  {27, 39}  accuracy is  0.85
Using feature(s)  {40, 27}  accuracy is  0.83
Feature set  {1, 27}  was best, accuracy is  0.96 %
Using feature(s)  {1, 2, 27}  accuracy is  0.94
Using feature(s)  {3, 1, 27}  accuracy is  0.93
Using feature(s)  {1, 27, 4}  accuracy is  0.93
Using feature(s)  {1, 27, 5}  accuracy is  0.92
Using feature(s)  {1, 27, 6}  accuracy is  0.92
Using feature(s)  {1, 27, 7}  accuracy is  0.93
Using feature(s)  {8, 1, 27}  accuracy is  0.94
Using feature(s)  {1, 27, 9}  accuracy is  0.94
Using feature(s)  {1, 10, 27}  accuracy is  0.92
```



Trace of Large Dataset Using Backward Elimination Feature Selection

```
-----------------------------------------------------------RUNNING BACKWARD SELECTION ON LARGE DATASET-----------------------------------------------------------
Welcome to the Feature Selection Algorithm.

Please enter total number of features: 40

1 - Forward Selection
2 - Backward Elimination
Type the number of the algorithm you want to run: 2

Beginning search.

Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 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]  accuracy is  0.69
Using feature(s)  [2, 3, 4, 5, 6, 7, 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]  accuracy is  0.74
Using feature(s)  [1, 3, 4, 5, 6, 7, 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]  accuracy is  0.71
Using feature(s)  [1, 2, 4, 5, 6, 7, 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]  accuracy is  0.67
Using feature(s)  [1, 2, 3, 5, 6, 7, 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]  accuracy is  0.68
Using feature(s)  [1, 2, 3, 4, 6, 7, 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]  accuracy is  0.69
Using feature(s)  [1, 2, 3, 4, 5, 7, 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]  accuracy is  0.7
Using feature(s)  [1, 2, 3, 4, 5, 6, 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]  accuracy is  0.71
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 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]  accuracy is  0.71
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 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]  accuracy is  0.68
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 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]  accuracy is  0.69
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 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]  accuracy is  0.7
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 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]  accuracy is  0.68
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 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]  accuracy is  0.66
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 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]  accuracy is  0.71
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 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]  accuracy is  0.7
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.7
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.68
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.67
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.7
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.68
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.69
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.7
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.71
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.71
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.68
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.68
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.85
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.66
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.7
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.71
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.7
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.7
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 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, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.7
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 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, 35, 36, 37, 38, 39, 40]  accuracy is  0.7
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 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, 36, 37, 38, 39, 40]  accuracy is  0.7
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 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, 37, 38, 39, 40]  accuracy is  0.71
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 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, 38, 39, 40]  accuracy is  0.7
Feature set  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  was best, accuracy is  0.85 %
Using feature(s)  [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.74
Using feature(s)  [1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.71
Using feature(s)  [1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.67
Using feature(s)  [1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.68
Using feature(s)  [1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.69
Using feature(s)  [1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.7
Using feature(s)  [1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.71
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.71
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.68
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.69
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.7
```

```
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.68
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.66
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.71
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.7
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.68
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.67
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.7
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.7
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.68
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.68
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.68
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.68
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.7
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.67
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.69
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.66
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.7
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.71
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 32, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.7
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 33, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.7
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 34, 35, 36, 37, 38, 39, 40]  accuracy is  0.7
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 35, 36, 37, 38, 39, 40]  accuracy is  0.68
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 34, 36, 37, 38, 39, 40]  accuracy is  0.7
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 35, 36, 37, 38, 39, 40]  accuracy is  0.68
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40]  accuracy is  0.68
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 37, 38, 39, 40]  accuracy is  0.68
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 38, 39, 40]  accuracy is  0.7
Using feature(s)  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 39, 40]  accuracy is  0.7
Warning! accuracy has decreased!So we stop searching..
Feature set  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]  was best, accuracy is  0.85 %

Feature Selection Elapsed Time: 75.00597929954529 seconds
Using features  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]

Classification Elapsed Time: 8.15431809425354 seconds
```