# Project 2 - Feature Selection with Nearest Neighbor

Yonas Adamu SID:

DATABASE ID: 110

***Note: Both forward and backward selections algorithm immediately stop when accuracy of current set {x} + N1… Nk drops below that of {x}.***

| Dataset | Best Features | Accuracy |
|---|---|---|
| **Small Dataset Number: 110** | Forward Selection = { 8, 10 } | 96% |
| | Backward Selection = { 2, 3, 6, 7, 8, 9, 10} | 89% |
| | Custom Algorithm = NA | |
| **Large Dataset Number: 110** | Forward Selection = { 23, 27} | 96.9% |
| | Backward Selection = REMOVED { 10, 24, 25, 28, 30, 35} | 77% |
| | Custom Algorithm = NA | |

*Reference:*

*Plot dataset using python*

**https://towardsdatascience.com/knn-visualization-in-just-13-lines-of-code-32820d72c6b6**

# I. Introduction

This project is about understanding Nearest Neighbor Classifier and its sensitivity to irrelevant features. For this project we are required to create a NN classifier model on a dataset which we will use to classify new data instances.

In addition, we are also required to make a leave-one-out validator that computes the accuracy of our NN classifier by comparing the actual class label of a single data instance of the training model with the result of the NN classifier for the same instance.

The NN classifier works on a set of features that is applied to classify a new data point from the training model. To accurately classify this new data point however we need to know which set of features we need to use. To do this, we need to select the best set of features that will result a higher accuracy for the NN classifier. For feature selection, we apply a forward-selection and backward selection algorithm that uses a greedy hill-climbing search.

# II. Challenges

There was no significant challenge for this project. However there was a little confusion regarding the feature selection algorithm. I had mixed up the greedy algorithm with a different algorithm where the first highest accuracy is taken without seeing the different set of features. However, an email from professor Sakellaridi put me back on track.

# III. Code Design

- ***Classifier*** class:
    - Reads the dataset file
    - Takes as input a set of features and an id for a single data instance in the dataset
    - Classifies the data point at id using euclidean distance to calculate its nearest neighbors and assigning the class label of the NN to the data point at id.

- ***Validator*** class:

    - Takes as input a set of features, the NN classifier and the entire dataset

    - Returns the accuracy of the on the given dataset when using the set of select features.

- ***Feature*** class:

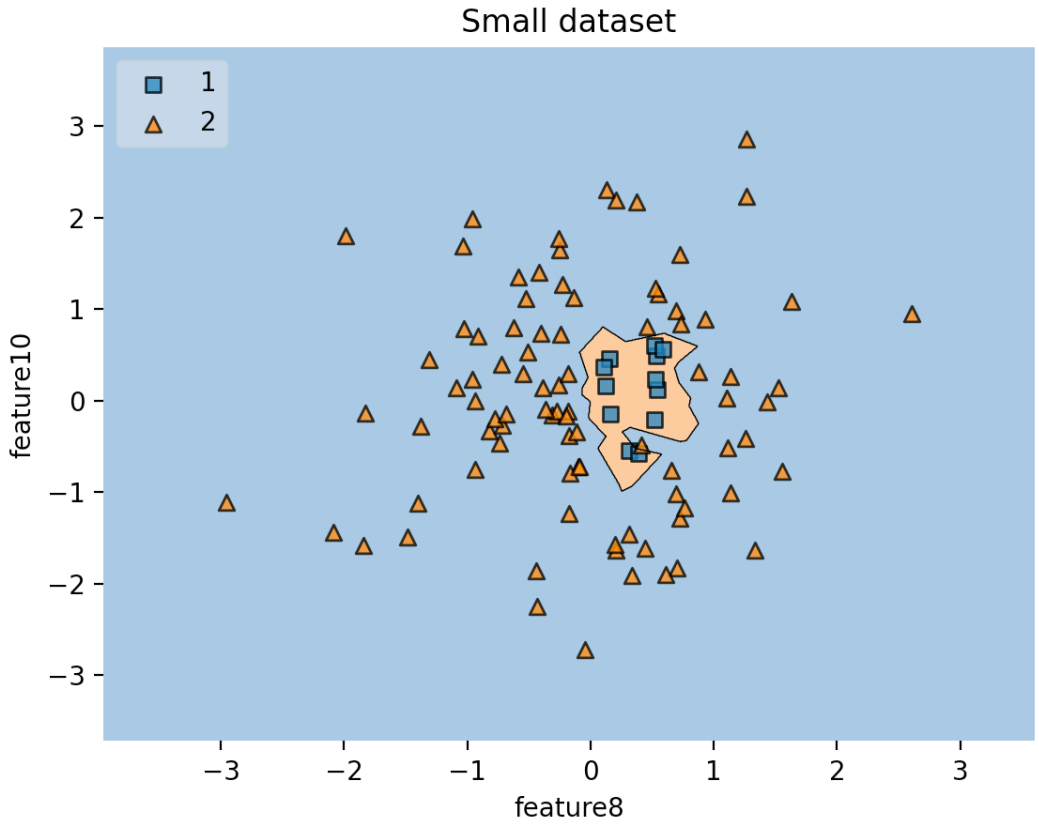
    - Takes as input a set of possible features,

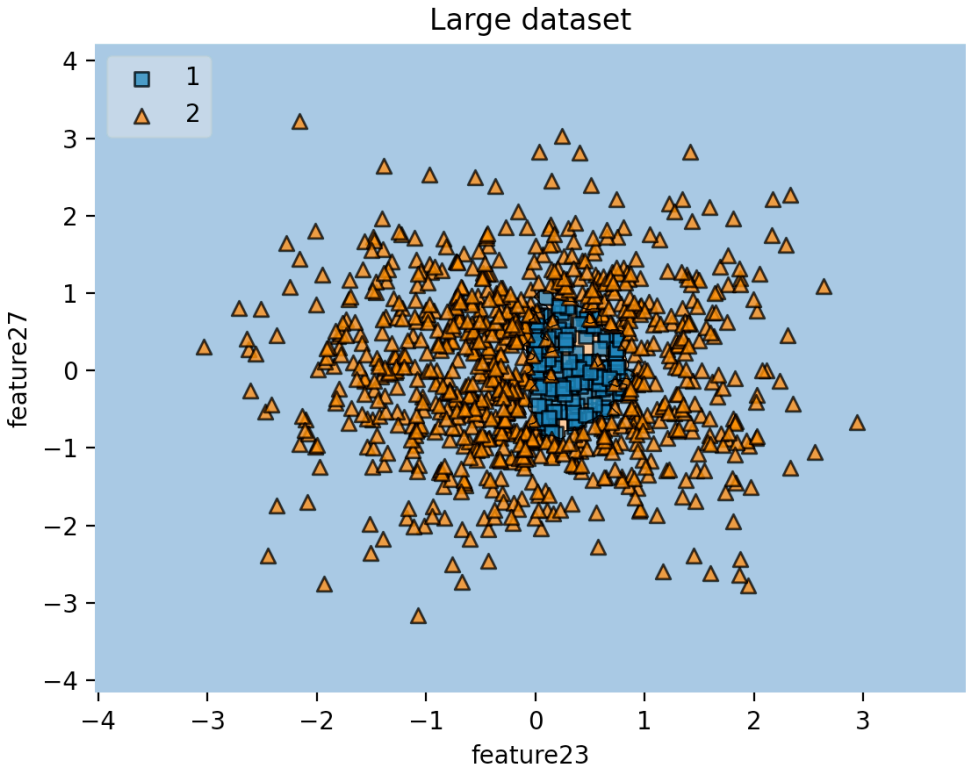
    - Outputs a subset of features that yield the highest accuracy

## IV. Dataset Details

- Small Dataset - 10 Number of features, 100 Number of instances
- Large Dataset - 40 Number of features, 1000 Number of instances

Plot using forward selection and



# V. Algorithms

### *1. Forward Selection: stops when accuracy drops*

The greedy forward feature selection begins by evaluating all feature subsets which consist of only one input feature. In other words, we use the k-fold cross validation to evaluate the accuracy of the one-component subsets, {X1}, {X2}, ..., {XM}, where M is the total number of the features, so that we can find the best individual feature, X1.

Next, it finds the best subset consisting of two components, X1 and one other feature from the remaining M-1 input features. if the current iteration results less accuracy than previous iteration stop.

### *2. Backward Selection: stops when accuracy drops*

For a possible number of features, we start with a set of all the features and remove a feature at a time and compare their accuracy. We select the one that yields the highest accuracy and add it to the set of features. We then continue the process until no more feature can be removed without decreasing the accuracy of the set of features.

### *3. Custom algorithm*

***Not Applicable***

# VI. Analysis

### *A. Experiment I - Forward Selection v backward selection*

- *No features selection v Feature Selection*

  Default rate(0 features selected):
  - small dataset: 88%
  - large Dataset: 84.5%

  No feature selection( **all features used**):
  - small dataset: 78%
  - large Dataset: 76.2%

  Feature Selection:
  - small dataset: 96%
  - large Dataset: 96.9%

  With no feature selection for both datasets, we are doing less than the default rate of the classifier. With feature selection however, we have a much higher accuracy and a better classifier model using select features.

- *Forward Selection v backward Selection*

  Forward Selection rate(0 features selected):
  - small dataset: 96% accuracy , 2 features
  - large Dataset: 96.7% accuracy, 2 features

Backward Selection:
small dataset: 89% accuracy , 7 features
large Dataset: 77% accuracy, 35 features

As we can see, Forward selection contains less features and higher accuracy than backward selection on both the large and small datasets.

- Pros and cons
  In terms of accuracy forward selection did much better than backward selection for both sets of datasets.
  In terms of time for completion, Forward selection was faster to conclude. (This could be due to style of coding.)

  Using the test dataset run and personal datasets, my analysis is that forward selection is better than backward selection as the forward selection resulted higher accuracy. In addition to having higher accuracy, the subset returned by forward selection contained fewer features than backward selection. This can be a very good thing as we have fewer features that could affect performance of our NN classifier negatively. In addition when we add time cost, forward selection seems better suited.

### *B. Experiment II - Effects of normalization*

Non normalized test on small dataset:
Backward selection
best feature(s): { 1, 2, 4, 5, 6, 7, 8, 10 } | accuracy: 81%
Forward selection
best feature(s): { 8, 10 } | accuracy: 97%

Comparing this with the normalized test found in the table at the top, accuracy dropped a little with normalized test on forward selection. On the other hand the accuracy of backward selection on normalized test showed a significant increase in accuracy. So we can say normalization has an effect on feature selection and NN classifier in general. But the effect can vary from one dataset to another.

### *C. Experiment III - Effects of the number of Nearest Neighbors (K)*

## VII. Conclusion

I think feature selection is perhaps the most important part of NN classification as it can have a significant effect on the accuracy of our classifier model. Both feature

selections that we saw have good merit on improving accuracy of our classification model. In terms of speed forward selection has a significant lead over backward selection as we start with no features and only observe small sets at a time while we consider the entire set of features for backward selection. If the best set of features are located at the start then forward selection does really well while backward selection will go thru a number of features to visit the same features.

## VIII. Trace of small Dataset

```
============================================================

Forward Selection on small dataset with: 10 features

-----------------------------------
{  } | 0.88%
-----------------------------------
On the 1-th level of the search tree:
  -- Using feature(s) {1} accuracy is 0.76%
  -- Using feature(s) {2} accuracy is 0.79%
  -- Using feature(s) {3} accuracy is 0.81%
  -- Using feature(s) {4} accuracy is 0.8%
  -- Using feature(s) {5} accuracy is 0.85%
  -- Using feature(s) {6} accuracy is 0.74%
  -- Using feature(s) {7} accuracy is 0.75%
  -- Using feature(s) {8} accuracy is 0.88%
  -- Using feature(s) {9} accuracy is 0.75%
  -- Using feature(s) {10} accuracy is 0.84%
  on level 1, feature 8 added to set {} with an accuracy of 0.88%

On the 2-th level of the search tree:
  -- Using feature(s) {8, 1} accuracy is 0.82%
  -- Using feature(s) {8, 2} accuracy is 0.85%
  -- Using feature(s) {8, 3} accuracy is 0.88%
  -- Using feature(s) {8, 4} accuracy is 0.89%
  -- Using feature(s) {8, 5} accuracy is 0.89%
  -- Using feature(s) {8, 6} accuracy is 0.8%
  -- Using feature(s) {8, 7} accuracy is 0.85%
  -- Using feature(s) {8, 9} accuracy is 0.87%
  -- Using feature(s) {8, 10} accuracy is 0.96%
  on level 2, feature 10 added to set {8} with an accuracy of 0.96%
```

On the 3-th level of the search tree:
  -- Using feature(s) {8, 10, 1} accuracy is 0.9%
  -- Using feature(s) {8, 10, 2} accuracy is 0.91%
  -- Using feature(s) {8, 10, 3} accuracy is 0.94%
  -- Using feature(s) {8, 10, 4} accuracy is 0.91%
  -- Using feature(s) {8, 10, 5} accuracy is 0.95%
  -- Using feature(s) {8, 10, 6} accuracy is 0.92%
  -- Using feature(s) {8, 10, 7} accuracy is 0.88%
  -- Using feature(s) {8, 10, 9} accuracy is 0.87%
  accuracy has decreased. Exiting

---
best feature(s) { 8, 10 } -> 0.96%
time spent: 0.682 seconds
==========

==========

Backward Selection on small dataset with: 10 features
{ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 } | 0.78%

On the 1-th level of the search tree:
  ---- removing feature 10 -> accuracy: 0.78%
  ---- removing feature 9 -> accuracy: 0.77%
  ---- removing feature 8 -> accuracy: 0.76%
  ---- removing feature 7 -> accuracy: 0.74%
  ---- removing feature 6 -> accuracy: 0.74%
  ---- removing feature 5 -> accuracy: 0.77%
  ---- removing feature 4 -> accuracy: 0.8%
  ---- removing feature 3 -> accuracy: 0.75%
  ---- removing feature 2 -> accuracy: 0.79%
  ---- removing feature 1 -> accuracy: 0.8%
  on level 1 feature 4 removed. set {1, 2, 3, 5, 6, 7, 8, 9, 10} -> 0.8%

On the 2-th level of the search tree:
  ---- removing feature 10 -> accuracy: 0.8%
  ---- removing feature 9 -> accuracy: 0.79%
  ---- removing feature 8 -> accuracy: 0.73%

---- removing feature 7 -> accuracy: 0.77%
---- removing feature 6 -> accuracy: 0.77%
---- removing feature 5 -> accuracy: 0.84%
---- removing feature 3 -> accuracy: 0.76%
---- removing feature 2 -> accuracy: 0.81%
---- removing feature 1 -> accuracy: 0.85%
on level 2 feature 1 removed. set {2, 3, 5, 6, 7, 8, 9, 10} -> 0.85%

On the 3-th level of the search tree:
---- removing feature 10 -> accuracy: 0.81%
---- removing feature 9 -> accuracy: 0.82%
---- removing feature 8 -> accuracy: 0.78%
---- removing feature 7 -> accuracy: 0.78%
---- removing feature 6 -> accuracy: 0.81%
---- removing feature 5 -> accuracy: 0.89%
---- removing feature 3 -> accuracy: 0.8%
---- removing feature 2 -> accuracy: 0.77%
on level 3 feature 5 removed. set {2, 3, 6, 7, 8, 9, 10} -> 0.89%

On the 4-th level of the search tree:
---- removing feature 10 -> accuracy: 0.81%
---- removing feature 9 -> accuracy: 0.86%
---- removing feature 8 -> accuracy: 0.86%
---- removing feature 7 -> accuracy: 0.86%
---- removing feature 6 -> accuracy: 0.82%
---- removing feature 3 -> accuracy: 0.88%
---- removing feature 2 -> accuracy: 0.82%
Accuracy has decreased. Exiting

---

best feature(s) { 2, 3, 6, 7, 8, 9, 10 } -> 0.89%
time spent: 2.228 seconds
==========