Project 2: Feature Selection with Nearest Neighbor

Student Name: Crystal Feng SID: 861285128

Solution: 22

Dataset	Best Feature Set	Accuracy
Small Number: 22	Forward Selection = {3, 9 , 6, 10}	0.92
	Backward Elimination = {2, 4, 6, 9, 10}	0.90
	Custom Algorithm = Not implemented	
Large Number: 22	Forward Selection = {35, 36}	0.979
	Backward Elimination = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39}	0.783
	Custom Algorithm = Not implemented	

----------<Begin Report>----------

In completing this project, I consulted following resources:

1. how to convert .txt data into matrix
   https://stackoverflow.com/questions/48833873/turn-txt-file-into-matrix-of-ints-in-python
2. how to 0 out columns in matrix
   https://stackoverflow.com/questions/36338927/how-to-zero-specific-column-of-2d-array
3. normalizing data
   https://developers.google.com/machine-learning/data-prep/transform/normalization

I. Introduction

In this project, we are tasked with finding the ideal feature sets to classify the given datasets. We utilize forward and backward search as well as k-nearest neighbors to compare the accuracies of different combinations of feature sets to find the best ones for classification.

The large datasets have more instances and features than the small datasets. Including the initial given datasets, we are also assigned specific datasets to use in this part of the project.

We started out designing the search algorithm first, then designed the code for classification and evaluation using the nearest neighbor algorithm. Once these parts were tested and verified, we combined the two to create a complete feature selection search.

II. Challenges

One of the biggest challenges to this project was the time it took to process the large data set. It takes around 20 mins for the search to complete on the large dataset. Each time I had to run the test on the large dataset, I was worried that it would crash before finishing. However, it always ended up finishing.

It was also my first time dealing with processing data from a .txt file. Thankfully there is plenty of documentation on built in libraries and functions that help with data parsing. With this I was able to use the datasets given and turn them into a useful data structure.

I also had trouble with the IDE I was using. My VSCode program would not recognize the paths for python libraries that I installed. This meant I could not use those functions in my program or import the necessary libraries. But they did work in a jupyter notebook instead of a plain .py file. I needed these libraries to create scatter plots and I was able to in the notebook.

III. Code Design

k is the number of features in the feature set

Data Prep

I first format the data into a parsable format. The data is given in a .txt file with data separated by lines and spaces. Each row represented a new instance separated by a newline; the spaces separated the columns. I created a helper function matrix() to format the data from the .txt file into a 2D array/matrix. Then the matrix is normalized by

column using the normalize() function I made. This data matrix is then returned to be used.

Forward Search

A forward search is used to iterate through the features in order to find the best feature set; this is done in my forward() function. We continuously add features to a set (increasing k) and use the current set to calculate the classification accuracy when calling the calcaccuracy() function. When increasing the number of features (k) decreases the accuracy, then we stop iterating and settle on the best set of features from the previous iteration with the best accuracy.

Backward Search

Backward search is like the opposite of forward search. We start by having the feature set include all the features, then every iteration, we exclude a feature from the set. This means decreasing the k. As we do in forward search, we keep track of the current best accuracy and the best accuracy at that k. Compare those accuracies and once decreasing the k decreases the accuracy, then we stop iterating. The best feature set is then found.

Accuracy Calculation and Classification

The calcaccuracy() function takes in the data matrix and feature set as parameters.

Looking at the feature set we are to analyze, we create a copy of the matrix with the features not in the feature set zeroed out. This is so that when comparing the euclidean distance between instances, those features are not used in the calculation.

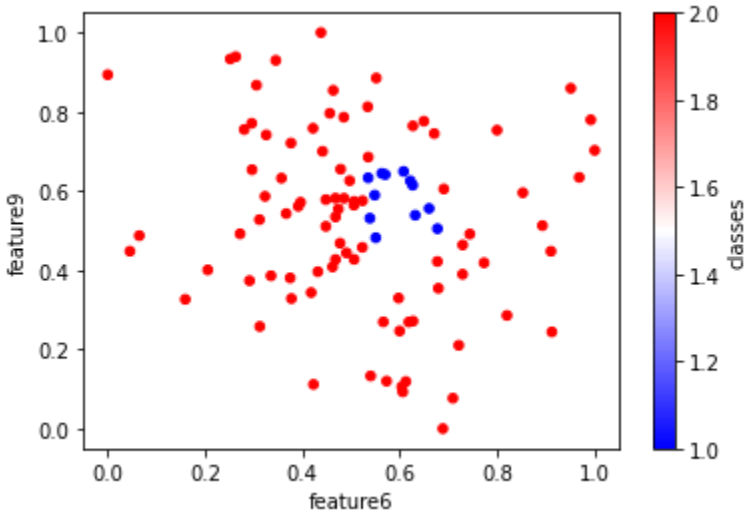
We then use all the instances but one for training. This one that is excluded is used to test the classification using nearest neighbor. The euclidean distance between the excluded instances and the rest of the instances is calculated, we use whichever instance is closest to the excluded instance to classify it. We know the correct classification and check to see if it has been classified correctly. If it has then we increase a counter that keeps track of the number of correct classifications. This is repeated with each instance in the dataset; we exclude one instance at a time for classification.

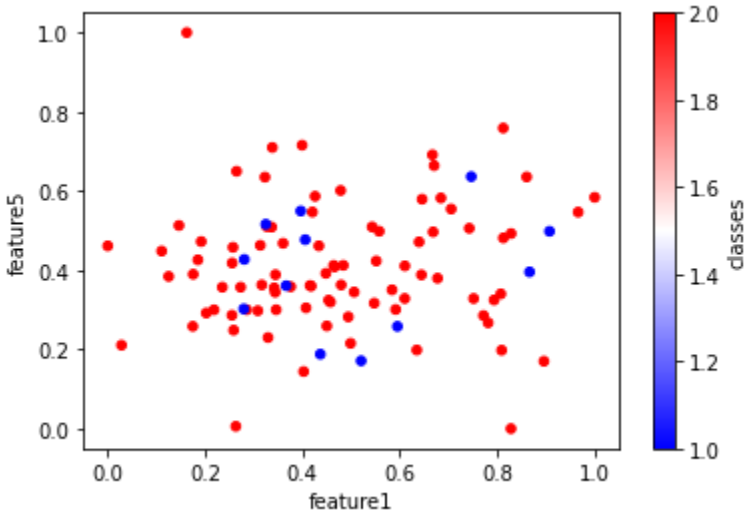
After all the instances have been classified, then we take that counter and divide it by the total number of instances to get the accuracy of the classification using that feature set.

IV. Dataset details

The dataset used for this analysis is the dataset personalized one assigned to me.

Your Small Dataset: 10 features, 100 instances



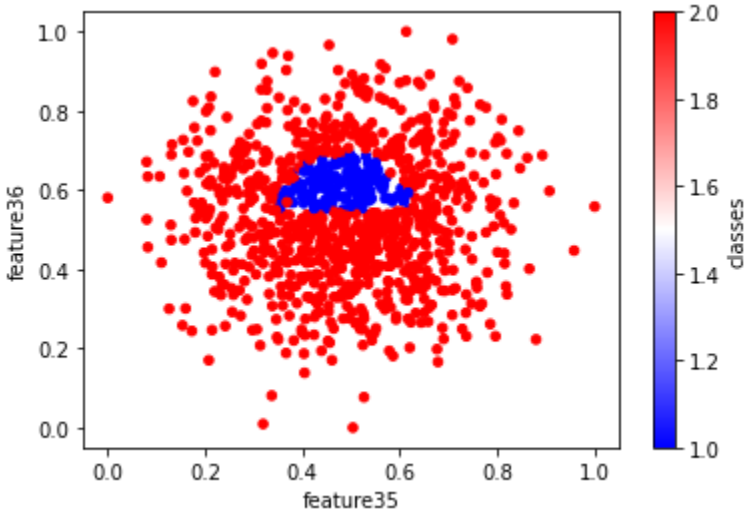


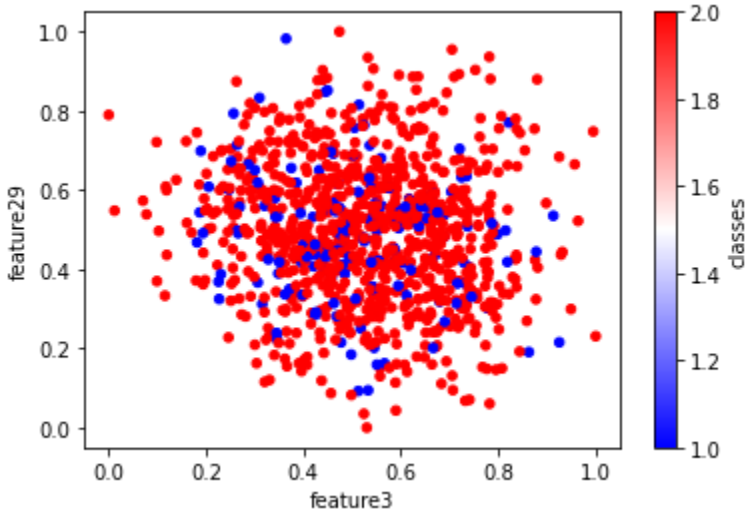
The figures above are two examples of plotting the small dataset.

The left figure shows good separation. You can distinctly see a separate of the classes with the blue points representing the instances of class 1 and red are the instances of class 2. **The accuracy using feature set [6, 9] has an accuracy of 95%.**

The right figure shows bad separation. Instances of class 1 and 2 are mixed and you can't make a good separated area of distinct classes. Unlike the left figure there is no distinct "grouping" of classes. **The accuracy using these features [1, 5] is 75%.** This is comparatively lower than the feature set with good separation.

Your Large Dataset: 40 of features, 1000 of instances





In the figures above, you can see examples of scatter plotting the large data set.

The left figure shows good separation. The instances of class 1 are grouped in the center of the graph while the instances of class 2 are clearly separated from the center group. **The accuracy using the feature set [35, 36] has an accuracy of 97.9%.**

The right figure shows bad separation. There is no clear separation between the red and blur dots. This means that the instances of class1 and 2 are not distinct using this feature set. **The accuracy using these features [3, 29] is 73.9%.** This is comparatively lower than the feature set with good separation.

V. Algorithms

1. Forward Selection
   In using forward selection, we are adding features to use as we go. You first start out with a single feature to use for classification. Then you compare the accuracies of those classifications. Choose the feature with the best accuracy to use and add another feature to make a feature set. This new feature set is then used to classify instances. Then again compare the accuracies of each feature set and choose the best one to expand on. We stop expanding when the accuracy starts to decrease.
   This way we are choosing the best feature set to expand on and adding features as we go.
2. Backward Elimination
   This is kind of the opposite of forward search, where we start with all the features and take off a single feature every time. This decrease in a single feature creates a new feature set. Compare the feature sets as be go, expanding on the set that has the best accuracy. As we do in forward search, we stop expanding when the accuracy starts to decrease.
   We are still choosing the best feature set to expand on, but unlike forward, we are excluding features as we expand.

VI. Analysis

Experiment 1: Comparing Forward Selection vs Backward Elimination.

Feature selection allows us to classify instances using features; not using features leaves us with a chance of classifying an instance correctly depending on how many instances of that class over total instances.

Example: A basket of fruits: 5 apples, 5 oranges. There are 10 fruits total so 10 instances. 5 instances of class apple and 5 instances of class orange. The chances of classifying an instance correctly is 5/10 = 50%. This is classification without observing any features of the fruits. We can't classify based on color, smell, or taste with no feature selection. But if we were able to select some features to use, our accuracy for classification would probably increase.

This idea can also be applied to our datasets. In the small data set, there are 100 total instances and 88 of them are of class 2. This means that there is an **accuracy of 88% with no feature selection.** But with feature selection, we are eventually able to get a **better accuracy of 90% using the feature set [3, 9 , 6, 10]**.

The main difference between forward and backward search is the starting feature set and how we expand. In forward search, we start with no features and expand by adding features to the set. For backward search, we start with all the features in the set and expand by taking out one feature at a time.

In both searches, we stop expanding once the accuracy decreases. This means that we are stopping at a local maxima of accuracy. If we continue the search either way, we might actually find a feature set with a higher accuracy, but to save on space and time, we stop. For forward search there might be a larger feature set with better accuracy; in backward search, there may exist a smaller feature set with better classification accuracy.

Maybe if we knew a range of how many feature sets is best, it'd make choosing between forward or backward easier. If we knew that a larger feature set would be more accurate then we'd use backward search or forward search for a smaller feature set.

If either search were to run through all combinations of features, they'd both give us the best accuracy in general and it'd be the same. But as it stands, we can see in the table on page 1 that this is not the case. The forward searches stop at smaller feature sets and backward search finds very large sets. Even though they are observing the same dataset, they give back different accuracies.

Backward search is good if we prefer looking at all the features first but it may take longer than forward search. Forward search is good if we want to look at very select features first.

Experiment 2: Effect of normalization

In preparing the data for classification, I store the data in a 2D array. Then I normalize feature columns. I do this by using the linear scaling formula, this scales the data to a normal range (0 to 1).

This should not affect the accuracy and classification that much since the data is still the same, just scaled differently. Similar to how 6/3 and 3 are the same thing. But there are slight changes when applying this to the project as seen in the traces below.

The left trace is without normalization, the right one is with normalization. We can observe the slight differences in accuracy using these feature sets. This trace is from my personalized large data set given. For example, looking at the feature set [37, 1, 27] the unnormalized accuracy is 94.8% while the normalized one is 94.6%. In the end, they both give me the same feature set with the best accuracy.

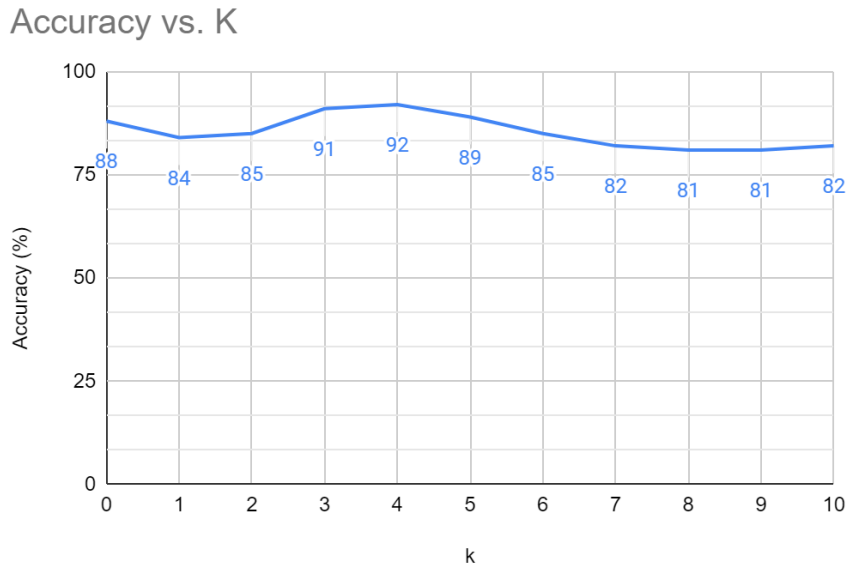
```
Using features(s) [37, 1, 27] accuracy is 94.8%

Using features(s) [38, 1, 27] accuracy is 93.5%

Using features(s) [39, 1, 27] accuracy is 93.89999999999999%

Using features(s) [40, 1, 27] accuracy is 92.80000000000001%
```

```
Using features(s) [37, 1, 27] accuracy is 94.6%

Using features(s) [38, 1, 27] accuracy is 93.5%

Using features(s) [39, 1, 27] accuracy is 93.89999999999999%

Using features(s) [40, 1, 27] accuracy is 92.9%
```

Experiment 3: Effect of number neighbors (k)

k refers to the number of features we use for classification. So if we are using a feature set [3, 5, 7] then k is 3 for the amount of features.

In the following graph, we can see how k can affect the accuracy of classification. The information used to plot the graph is from the small dataset. I allowed the forward search to continue without stopping when accuracy drops. Then used the best of each k to plot this graph.



As we can see, the graph with the highest accuracy is when we use 4 features. Then accuracy starts to decrease after that. There are also local maximas at k = 0 and k = 10. If there had been a local maxima at k = 1 and it decreased at k = 2 then our search would have returned a feature set with k of 1 as the best for classification.

VII. Conclusion

Our exploration of classification and search allowed us to see the trends and usefulness of k-NN classification. Further analysis lets us see how and when k-NN works best and how the accuracy relates to classification accuracy. We were also able to see when we should use forward and backward search.

Forward and backward search both have their uses, and we can determine which one we want to use depending on what kind of data set we want to find. When dealing with forward search, we work with less features at first with a smaller k. We then iteratively add to the set and therefore increase k. In backward search, we start out with all the features and so the max k. We iteratively decrease the amount of features in turn decreasing k. Depending on the approach we want to take, we could use either search methods.

Normalizing our data allows us to transform our features to be on a similar scale and potentially improves performance and training stability of the model (3). As we saw earlier, by normalizing our data, there are small differences in accuracy. This is most apparent in the large data set than in the small one.

In our earlier exploration of the dataset, we can see the correlation between accuracy and good separation. Good separation between the classes using the right features meant better classification accuracy. The classes were grouped better than others. The features that did not group the classes well had a worse accuracy for classification.

One of the challenges I faced was the long computing time on the large dataset, so one thing I could improve of this project is a better search algorithm to cut down on processing time.

This project allowed us to observe the different aspects of search and k-NN classification: from normalization to forward and backward search. We were able to analyze and find correlations between our data.

VIII. Trace of your small dataset

Welcome to Crystal Feng's Evaluation Program.

Beginning Search

This dataset has 10 features (not including the class attribute), with 100 instances.

Please wait while I normalize the data ...

Done!

Running nearest neighbor with no features (default rate), using "leave on out" evaluation, I get an accuracy of 88.0%

Using features(s) [1] accuracy is 76.0%

Using features(s) [2] accuracy is 77.0%

Using features(s) [3] accuracy is 76.0%

Using features(s) [4] accuracy is 83.0%

Using features(s) [5] accuracy is 76.0%

Using features(s) [6] accuracy is 77.0%

Using features(s) [7] accuracy is 77.0%

Using features(s) [8] accuracy is 78.0%

Using features(s) [9] accuracy is 80.0%

Using features(s) [10] accuracy is 84.0%

Feature set [10] was best, accuracy is 84.0%

Using features(s) [1, 10] accuracy is 77.0%

Using features(s) [2, 10] accuracy is 81.0%

Using features(s) [3, 10] accuracy is 80.0%

Using features(s) [4, 10] accuracy is 80.0%

Using features(s) [5, 10] accuracy is 78.0%

Using features(s) [6, 10] accuracy is 85.0%

Using features(s) [7, 10] accuracy is 75.0%

Using features(s) [8, 10] accuracy is 77.0%

Using features(s) [9, 10] accuracy is 82.0%

Feature set [6, 10] was best, accuracy is 85.0%

Using features(s) [1, 6, 10] accuracy is 80.0%
Using features(s) [2, 6, 10] accuracy is 82.0%
Using features(s) [3, 6, 10] accuracy is 80.0%
Using features(s) [4, 6, 10] accuracy is 79.0%
Using features(s) [5, 6, 10] accuracy is 83.0%
Using features(s) [7, 6, 10] accuracy is 83.0%
Using features(s) [8, 6, 10] accuracy is 78.0%
Using features(s) [9, 6, 10] accuracy is 91.0%
Feature set [9, 6, 10] was best, accuracy is 91.0%

Using features(s) [1, 9, 6, 10] accuracy is 85.0%
Using features(s) [2, 9, 6, 10] accuracy is 88.0%
Using features(s) [3, 9, 6, 10] accuracy is 92.0%
Using features(s) [4, 9, 6, 10] accuracy is 85.0%
Using features(s) [5, 9, 6, 10] accuracy is 87.0%
Using features(s) [7, 9, 6, 10] accuracy is 86.0%
Using features(s) [8, 9, 6, 10] accuracy is 86.0%
Feature set [3, 9, 6, 10] was best, accuracy is 92.0%

Using features(s) [1, 3, 9, 6, 10] accuracy is 81.0%
Using features(s) [2, 3, 9, 6, 10] accuracy is 89.0%
Using features(s) [4, 3, 9, 6, 10] accuracy is 78.0%
Using features(s) [5, 3, 9, 6, 10] accuracy is 84.0%
Using features(s) [7, 3, 9, 6, 10] accuracy is 84.0%
Using features(s) [8, 3, 9, 6, 10] accuracy is 88.0%

(Warning, Accuracy has decreased!!)
Finished search!! The best feature subset is [3, 9, 6, 10] which has an accuracy of 92.0%