Duke Pham
CS170 Spring 2022
Project 2
SID: 861163782

In completing this project, I consulted some of the following resources:

https://www.geeksforgeeks.org/python-check-if-a-file-or-directory-exists-2/?ref=lbp

https://www.geeksforgeeks.org/import-text-files-into-numpy-arrays/

https://www.programiz.com/python-programming/shallow-deep copy#:~:text=%20Python%20Shallow%20Copy%20and%20Deep%20Copy%20,stores%20the%20reference%20of%20the%20original...%20More%20

https://stackoverflow.com/questions/35109113/how-to-normalize-only-certain-columns-in-scikit-learn

Introduction

Neural networks is perhaps the most interesting subset of Artificial Intelligence we can learn from. For this project, we implemented two different search algorithms which were Forward Selection and Backward Elimination. There is also the implementation of using the leave-one out-validation method and the Nearest Neighbor classifier. This project was perhaps the most challenging. This report will summarize some of the details in getting the project to work.

Challenges

Some of the challenges that I've come across were doing the coding implementation in C++ and switching to Python language. Particularly, I was having a difficult time in converting the text file dataset into a 2D array and reading for the correct input. Another challenge was understanding the two-search algorithm and being able implement them for the whole project to work properly. If I was able to see the two discussion slides provided I think would have a better experience completing the project. In addition, I was facing numerous errors with Python particular with TabErrors.

Code Design

```
import numpy as np
from sklearn.preprocessing import normalize
import os
import copy

## forward Selection funtion
def forwardSelection(data, maxSet, maxAccuracy):
    row, column = data.shape
    bestAccuracy = maxAccuracy
    currentSet = []
    for i in range(1, column):
        addFeature = 0
        currAccuracy = 0
        for j in maxSet:
            if not j in currentSet:
                tempSet = currentSet + [j]
```

Duke Pham
CS170 Spring 2022
Project 2
SID: 861163782

```
                accuracy = oneoutValidator(data, tempSet, row)
                #print("Using feature(s) " + str(currentSet)
                #           + ", accuracy is " + str(currAccuracy) + "%\n")
                if accuracy > currAccuracy:
                    currAccuracy = accuracy
                    addFeature = j

        currentSet.append(addFeature)
        print("Best accuracy are feature(s) " + str(currentSet) + ", accuracy is
" + "{0:.1f}".format(currAccuracy) + "%\n")

        if currAccuracy > bestAccuracy:
            bestAccuracy = currAccuracy
            bestSet = copy.deepcopy(currentSet)
        elif currAccuracy < bestAccuracy:
            print("(Warning, Accuracy has decreased! "
                        + "Continuing search in case of local maxima)\n")
            ##print("Using feature(s) " + str(bestSet) + " accuracy is " +
str(accuracy) + "%\n")
    return bestSet, bestAccuracy

## backward elimination function
def backwardElimination(data, maxSet, maxAccuracy):
    row, column = data.shape
    bestAccuracy = maxAccuracy
    currentSet = [1,2,3,4,5,6,7,8,9,10]
    for i in range(1, column):
        addFeature = 0
        currAccuracy = 0
        for j in maxSet:
            if j in currentSet:
                tempSet = currentSet + []
                tempSet.remove(j)
                accuracy = oneoutValidator(data, tempSet, row)
                #print("Using feature(s) " + str(currentSet)
                #           + ", accuracy is " + str(currAccuracy) + "%\n")
                if accuracy > currAccuracy:
                    currAccuracy = accuracy
                    addFeature = j
        currentSet.remove(addFeature)
        print("Best accuracy are feature(s) " + str(currentSet) + ", accuracy is
" + "{0:.1f}".format(currAccuracy) + "%\n")
        if currAccuracy > bestAccuracy:
```

Duke Pham
CS170 Spring 2022
Project 2
SID: 861163782

```
            bestAccuracy = currAccuracy
            bestSet = copy.deepcopy(currentSet)
        elif currAccuracy < bestAccuracy:
            print("(Warning, Accuracy has decreased! "
                        + "Continuing search in case of local maxima)\n")
            ##print("Using feature(s) " + str(bestSet) + " accuracy is " +
str(accuracy) + "%\n")
    return bestSet, bestAccuracy

## classifier function
def nnClassifier(data, datapoint, subFeature, numInstances):
    nearestneighbor = 0
    shortestDist = float('inf')
    for i in range(numInstances):
        if (datapoint != i):
            distance = 0
            for j in subFeature:
                distance = distance + pow(
                    (data[i][j] - data[datapoint][j]),
                    2)
            distance = pow(distance, 0.5)
            if distance < shortestDist:
                nearestneighbor = i
                shortestDist = distance
    return nearestneighbor

## evaluation function
def oneoutValidator(data, subFeatures, numInstances):
    row, column = data.shape
    correctInstances = 0
    for i in range(numInstances):
        leaveOne = i
        neighbor = nnClassifier(data, leaveOne, subFeatures, numInstances)
        if data[neighbor][0] == data[leaveOne][0]:
            correctInstances = correctInstances + 1
    accuracy = (correctInstances / (numInstances-1)) * 100
    print("Using feature(s) " + str(subFeatures) + ", accuracy is " +
"{0:.1f}".format(accuracy) + "%")
    return accuracy
```

Duke Pham
CS170 Spring 2022
Project 2
SID: 861163782

```
## Main
#if __name__ == "__main__":
print("Welcome to Duke Pham Feature Selection Algorithm.")
filename = input("Type in the name of the file to test: ")
if not os.path.isfile(filename):
        print("Error: File does not exist")
        exit(1)

## load in txt file and convert to 2D array
array = np.loadtxt(filename)


row, column = array.shape
## set the number of features and instances
features = column - 1
instances = row

fullFeature = list(range(1, column))
normalized = copy.deepcopy(array)
normalized = normalize(normalized[:, fullFeature], axis = 0)
array = np.concatenate((array[:, [0]], normalized), axis=1)


accuracy = oneoutValidator(array, fullFeature, instances)

algoChoice = input("Type the number of the algorithm you want to run.\n" +
                                    "1: Forward Selection\n" + "2: Backward
Elimination\n" +
                                    "3: Duke's Special Algorithm.\n")
if algoChoice != "1" and algoChoice != "2" and algoChoice != "3":
        print("Error: Incorrect choice of algorithm entered")
        exit(1)

print("This dataset has " + str(features) + " features" +
           " (not including the class attributes)" + ", with " +
           str(instances) + " instances.\n")

print("Running the nearest neighbor with no features (default rate)," +
           " using 'leaving-one-out' evaluation," + " I get an accuracy of " +
           str(accuracy) + "%")
print("\n")
print("Beginning Search\n")
```

Duke Pham
CS170 Spring 2022
Project 2
SID: 861163782

```
if algoChoice == "1":
        print("forward Selection")
        bestSet, bestAccuracy = forwardSelection(array, fullFeature, accuracy)
elif algoChoice == "2":
        print("backward Elimination")
        bestSet, bestAccuracy = backwardElimination(array, fullFeature,

              accuracy)


print("Finished Search!! The best feature subset is " + str(bestSet) +
                    ", which has an accuracy of " +
"{0:.1f}".format(bestAccuracy) + "%")
```

Dataset Detail

small-test-dataset.txt

40 features, 1000 instances

Large-test-dataset.txt

10 features, 100 instances

Algorithms

Forward Selection:

Start an empty feature set and adds a new relevant feature until it reaches its goal

Backward Elimination:

Starts from a full feature set and removes the least relevant feature until it reaches its

Analysis

Experiment1: Comparing Forward Selection vs Backward Elimination

Having no feature selection is more accurate than having a feature selection. It would take some time for the machine to create a tree and search through that tree and determine the possible paths. With feature selection, we would sacrifice some amount of accuracy in the exchange of faster processing.

Comparing the feature set and accuracy for forward selection vs backward selection. I observed how forward selection will give us a short feature set compared to backward elimination. With data processing, forward selection would give us a higher accuracy compared to backward elimination.

Duke Pham
CS170 Spring 2022
Project 2
SID: 861163782

Between the pros and cons with both algorithms, backward elimination seems to be fast but not as accurate as forward selection. Forward selection won't be as fast but very accurate.

Conclusion

In conclusion, I was able to do the project again and be able to make some analysis. I find that backward elimination is faster than forward selection. In experimenting with the small dataset, the best feature set for forward selection is [5,3] which has an accuracy 92.9% accuracy. For backward elimination, the best feature set is [5,10] which has an accuracy of 82.8%.

Trace of your small dataset

Welcome to Duke Pham Feature Selection Algorithm.

Type in the name of the file to test: small-test-dataset.txt

Using feature(s) [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], accuracy is 66.7%

Type the number of the algorithm you want to run.

1: Forward Selection

2: Backward Elimination

3: Duke's Special Algorithm.

1

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

Running the nearest neighbor with no features (default rate), using 'leaving-one-out' evaluation, I get an accuracy of 66.66666666666666%

Beginning Search

forward Selection

Using feature(s) [1], accuracy is 57.6%

Using feature(s) [2], accuracy is 54.5%

Using feature(s) [3], accuracy is 68.7%

Using feature(s) [4], accuracy is 65.7%

Duke Pham
CS170 Spring 2022
Project 2
SID: 861163782

Using feature(s) [5], accuracy is 75.8%

Using feature(s) [6], accuracy is 61.6%

Using feature(s) [7], accuracy is 62.6%

Using feature(s) [8], accuracy is 60.6%

Using feature(s) [9], accuracy is 66.7%

Using feature(s) [10], accuracy is 64.6%

Best accuracy are feature(s) [5], accuracy is 75.8%

Using feature(s) [5, 1], accuracy is 76.8%

Using feature(s) [5, 2], accuracy is 80.8%

Using feature(s) [5, 3], accuracy is 92.9%

Using feature(s) [5, 4], accuracy is 75.8%

Using feature(s) [5, 6], accuracy is 79.8%

Using feature(s) [5, 7], accuracy is 80.8%

Using feature(s) [5, 8], accuracy is 77.8%

Using feature(s) [5, 9], accuracy is 73.7%

Using feature(s) [5, 10], accuracy is 82.8%

Best accuracy are feature(s) [5, 3], accuracy is 92.9%

Using feature(s) [5, 3, 1], accuracy is 81.8%

Using feature(s) [5, 3, 2], accuracy is 82.8%

Using feature(s) [5, 3, 4], accuracy is 84.8%

Using feature(s) [5, 3, 6], accuracy is 82.8%

Using feature(s) [5, 3, 7], accuracy is 90.9%

Using feature(s) [5, 3, 8], accuracy is 79.8%

Using feature(s) [5, 3, 9], accuracy is 84.8%

Using feature(s) [5, 3, 10], accuracy is 85.9%

Duke Pham
CS170 Spring 2022
Project 2
SID: 861163782

Best accuracy are feature(s) [5, 3, 7], accuracy is 90.9%

(Warning, Accuracy has decreased! Continuing search in case of local maxima)

Using feature(s) [5, 3, 7, 1], accuracy is 87.9%

Using feature(s) [5, 3, 7, 2], accuracy is 79.8%

Using feature(s) [5, 3, 7, 4], accuracy is 82.8%

Using feature(s) [5, 3, 7, 6], accuracy is 87.9%

Using feature(s) [5, 3, 7, 8], accuracy is 81.8%

Using feature(s) [5, 3, 7, 9], accuracy is 82.8%

Using feature(s) [5, 3, 7, 10], accuracy is 84.8%

Best accuracy are feature(s) [5, 3, 7, 1], accuracy is 87.9%

(Warning, Accuracy has decreased! Continuing search in case of local maxima)

Using feature(s) [5, 3, 7, 1, 2], accuracy is 79.8%

Using feature(s) [5, 3, 7, 1, 4], accuracy is 77.8%

Using feature(s) [5, 3, 7, 1, 6], accuracy is 86.9%

Using feature(s) [5, 3, 7, 1, 8], accuracy is 75.8%

Using feature(s) [5, 3, 7, 1, 9], accuracy is 76.8%

Using feature(s) [5, 3, 7, 1, 10], accuracy is 73.7%

Best accuracy are feature(s) [5, 3, 7, 1, 6], accuracy is 86.9%

(Warning, Accuracy has decreased! Continuing search in case of local maxima)

Using feature(s) [5, 3, 7, 1, 6, 2], accuracy is 74.7%

Using feature(s) [5, 3, 7, 1, 6, 4], accuracy is 73.7%

Duke Pham
CS170 Spring 2022
Project 2
SID: 861163782

Using feature(s) [5, 3, 7, 1, 6, 8], accuracy is 77.8%

Using feature(s) [5, 3, 7, 1, 6, 9], accuracy is 70.7%

Using feature(s) [5, 3, 7, 1, 6, 10], accuracy is 71.7%

Best accuracy are feature(s) [5, 3, 7, 1, 6, 8], accuracy is 77.8%

(Warning, Accuracy has decreased! Continuing search in case of local maxima)

Using feature(s) [5, 3, 7, 1, 6, 8, 2], accuracy is 70.7%

Using feature(s) [5, 3, 7, 1, 6, 8, 4], accuracy is 70.7%

Using feature(s) [5, 3, 7, 1, 6, 8, 9], accuracy is 68.7%

Using feature(s) [5, 3, 7, 1, 6, 8, 10], accuracy is 73.7%

Best accuracy are feature(s) [5, 3, 7, 1, 6, 8, 10], accuracy is 73.7%

(Warning, Accuracy has decreased! Continuing search in case of local maxima)

Using feature(s) [5, 3, 7, 1, 6, 8, 10, 2], accuracy is 66.7%

Using feature(s) [5, 3, 7, 1, 6, 8, 10, 4], accuracy is 71.7%

Using feature(s) [5, 3, 7, 1, 6, 8, 10, 9], accuracy is 65.7%

Best accuracy are feature(s) [5, 3, 7, 1, 6, 8, 10, 4], accuracy is 71.7%

(Warning, Accuracy has decreased! Continuing search in case of local maxima)

Using feature(s) [5, 3, 7, 1, 6, 8, 10, 4, 2], accuracy is 61.6%

Using feature(s) [5, 3, 7, 1, 6, 8, 10, 4, 9], accuracy is 64.6%

Best accuracy are feature(s) [5, 3, 7, 1, 6, 8, 10, 4, 9], accuracy is 64.6%

(Warning, Accuracy has decreased! Continuing search in case of local maxima)

Duke Pham
CS170 Spring 2022
Project 2
SID: 861163782

Using feature(s) [5, 3, 7, 1, 6, 8, 10, 4, 9, 2], accuracy is 66.7%

Best accuracy are feature(s) [5, 3, 7, 1, 6, 8, 10, 4, 9, 2], accuracy is 66.7%

(Warning, Accuracy has decreased! Continuing search in case of local maxima)

Finished Search!! The best feature subset is [5, 3], which has an accuracy of 92.9%