CS170 Introduction to AI Project 2, Part 1

Teammate 1: Edwin Leon Du SID: 862132870
Teammate 2: Josh McIntyre SID: 862054277

Code

GitHub: https://github.com/EdwinLeon98/CS170_Project2

```
import random

# Node class
class Node():

    def __init__(self, f):
        self.feats = set()
        for i in f:
            self.feats.add(i)
        self.acc = round(random.uniform(1.00, 100.00), 2)

print("Welcome to Edwin Leon and Josh McIntyre's Feature Selection
Algorithm")

# Features prompt and input sanitization
features = None
invalid = True
while invalid:
    features = input("Please enter the total number of features\n")
    if int(features) < 0:
        invalid = True
    else:
        invalid = False

# Algorithm prompt and input sanitization
algorithm = None
invalid = True
while invalid:
    algorithm = input("Type the number of the algorithm you want to
run\n1. Forward Selection\n2. Backward Elimination\n")
    if(algorithm == '1'):
        invalid = False
    elif(algorithm == '2'):
        invalid = False
    else:
        invalid = True
```

```
# Forward Selection Search
if(algorithm == '1'):
    n = Node(set())         # Set n to Node with features = {}
    print("Using no features and \"random\" evaluation, we get an
accuracy of {}%".format(n.acc))
    print("Beginning search.")
    currMax = n             # Set current and true max to n
    trueMax = currMax
    size = 1
    decreased = False       # Flag to see if our accuracy decreased
during an iteration of our search

    while not size > int(features):
        tmpMax = currMax    # tmpMax helps track last iterations max
Node
        count = 1           # helps track current iteration's first
valid subset

        # Loop to find currMax
        for i in range(1, int(features)+1):
            n1 = Node(set.union({i}, tmpMax.feats))     # tmpMax
holds last iteration's subset

            # Do not want to create subsets we checked last iteration
            if not len(n1.feats) == size:
                continue

            # Init currMax to first valid subset
            if count == 1:
                currMax = n1

            # Updates currMax
            if n1.acc >= currMax.acc:
                currMax = n1
            print("Using feature(s) {} accuracy is
{}%".format(str(n1.feats), n1.acc))
            count += 1

        if not size+1 > int(features):
            print("\nFeature set {} was best, accuracy is
{}%\n".format(str(currMax.feats), currMax.acc))

        # Update true max, and set decreased flag if currMax <
trueMax, meaning our search decreased from a higher accuracy
        if currMax.acc >= trueMax.acc:
            trueMax = currMax

        if currMax.acc < tmpMax.acc:
            print("(Warning, Accuracy has decreased! Continuing
search in case of local maxima)\n")
        size += 1
```

```
    # Display results
    if len(trueMax.feats) == 0:
        print("Finished search!! The best feature subset is {{}},
which has an accuracy of {}%".format(trueMax.acc))
    else:
        print("Finished search!! The best feature subset is {}, which
has an accuracy of {}%".format(str(trueMax.feats), trueMax.acc))

# Backward Elimination Search
else:
    n = Node(set())         # Set n to Node with features = {1, 2, 3,
4, ..., nFeatures}
    for i in range(1, int(features)+1):
        n.feats.add(i)
    print("Using all features {} and \"random\" evaluation, we get an
accuracy of {}%".format(n.feats, n.acc))
    print("Beginning search.")
    currMax = n             # Set current and true max to n
    trueMax = currMax
    size = int(features)-1
    decreased = False       # Flag to see if our accuracy decreased
during an iteration of our search

    while not size < 0:
        tmpMax = currMax    # tmpMax helps track last iterations max
Node
        count = 1           # helps track current iteration's first
valid subset

        # Loop to find currMax
        for i in range(1, int(features)+1):
            n1 = Node(tmpMax.feats)     # tmpMax holds last
iteration's subset
            if(i in n1.feats):
                n1.feats.remove(i)

            # Do not want to create subsets we checked last iteration
            if not len(n1.feats) == size:
                continue

            # Init currMax to first valid subset
            if count == 1:
                currMax = n1

            # Updates currMax
            if n1.acc >= currMax.acc:
                currMax = n1
            if size == 0:
                print("Using feature(s) {{}} accuracy is
{}%".format(n1.acc))
            else:
```

```
                print("Using feature(s) {} accuracy is
{}%".format(str(n1.feats), n1.acc))
            count += 1

        if not size == 0:
            print("\nFeature set {} was best, accuracy is
{}%\n".format(str(currMax.feats), currMax.acc))

        # Update true max, and set decreased flag if currMax <
trueMax, meaning our search decreased from a higher accuracy
        if currMax.acc >= trueMax.acc:
            trueMax = currMax

        if currMax.acc < tmpMax.acc:
            print("(Warning, Accuracy has decreased! Continuing
search in case of local maxima)\n")
        size -= 1

    # Display results
    if len(trueMax.feats) == 0:
        print("Finished search!! The best feature subset is {{}},
which has an accuracy of {}%".format(trueMax.acc))
    else:
        print("Finished search!! The best feature subset is {}, which
has an accuracy of {}%".format(str(trueMax.feats), trueMax.acc))
```

Traces

Forward Selection

```
Welcome to Edwin Leon and Josh McIntyre's Feature Selection Algorithm
Please enter the total number of features
4
Type the number of the algorithm you want to run
1. Forward Selection
2. Backward Elimination
1
Using no features and "random" evaluation, we get an accuracy of 48.03%
Beginning search.
Using feature(s) {1} accuracy is 43.77%
Using feature(s) {2} accuracy is 40.77%
Using feature(s) {3} accuracy is 84.54%
Using feature(s) {4} accuracy is 3.46%

Feature set {3} was best, accuracy is 84.54%

Using feature(s) {1, 3} accuracy is 84.3%
Using feature(s) {2, 3} accuracy is 62.07%
Using feature(s) {3, 4} accuracy is 27.48%

Feature set {1, 3} was best, accuracy is 84.3%

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

Using feature(s) {1, 2, 3} accuracy is 56.63%
Using feature(s) {1, 3, 4} accuracy is 60.0%

Feature set {1, 3, 4} was best, accuracy is 60.0%

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

Using feature(s) {1, 2, 3, 4} accuracy is 15.03%
(Warning, Accuracy has decreased! Continuing search in case of local maxima)

Finished search!! The best feature subset is {3}, which has an accuracy of 84.54%
```

```
Welcome to Edwin Leon and Josh McIntyre's Feature Selection Algorithm
Please enter the total number of features
6
Type the number of the algorithm you want to run
1. Forward Selection
2. Backward Elimination
1
Using no features and "random" evaluation, we get an accuracy of 36.31%
Beginning search.
Using feature(s) {1} accuracy is 16.2%
Using feature(s) {2} accuracy is 82.98%
Using feature(s) {3} accuracy is 81.39%
Using feature(s) {4} accuracy is 37.0%
Using feature(s) {5} accuracy is 19.47%
Using feature(s) {6} accuracy is 52.08%

Feature set {2} was best, accuracy is 82.98%

Using feature(s) {1, 2} accuracy is 48.45%
Using feature(s) {2, 3} accuracy is 37.94%
Using feature(s) {2, 4} accuracy is 75.76%
Using feature(s) {2, 5} accuracy is 14.49%
Using feature(s) {2, 6} accuracy is 35.65%

Feature set {2, 4} was best, accuracy is 75.76%

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

Using feature(s) {1, 2, 4} accuracy is 27.45%
Using feature(s) {2, 3, 4} accuracy is 6.72%
Using feature(s) {2, 4, 5} accuracy is 75.85%
Using feature(s) {2, 4, 6} accuracy is 87.37%

Feature set {2, 4, 6} was best, accuracy is 87.37%

Using feature(s) {1, 2, 4, 6} accuracy is 5.58%
Using feature(s) {2, 3, 4, 6} accuracy is 7.58%
Using feature(s) {2, 4, 5, 6} accuracy is 80.76%

Feature set {2, 4, 5, 6} was best, accuracy is 80.76%

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

Using feature(s) {1, 2, 4, 5, 6} accuracy is 48.91%
Using feature(s) {2, 3, 4, 5, 6} accuracy is 83.42%

Feature set {2, 3, 4, 5, 6} was best, accuracy is 83.42%

Using feature(s) {1, 2, 3, 4, 5, 6} accuracy is 65.44%
(Warning, Accuracy has decreased! Continuing search in case of local maxima)

Finished search!! The best feature subset is {2, 4, 6}, which has an accuracy of 87.37%
```

Backward Elimination

```
Welcome to Edwin Leon and Josh McIntyre's Feature Selection Algorithm
Please enter the total number of features
4
Type the number of the algorithm you want to run
1. Forward Selection
2. Backward Elimination
2
Using all features {1, 2, 3, 4} and "random" evaluation, we get an accuracy of 54.17%
Beginning search.
Using feature(s) {2, 3, 4} accuracy is 55.22%
Using feature(s) {1, 3, 4} accuracy is 40.34%
Using feature(s) {1, 2, 4} accuracy is 43.65%
Using feature(s) {1, 2, 3} accuracy is 10.27%

Feature set {2, 3, 4} was best, accuracy is 55.22%

Using feature(s) {3, 4} accuracy is 65.25%
Using feature(s) {2, 4} accuracy is 77.95%
Using feature(s) {2, 3} accuracy is 39.0%

Feature set {2, 4} was best, accuracy is 77.95%

Using feature(s) {4} accuracy is 65.01%
Using feature(s) {2} accuracy is 85.31%

Feature set {2} was best, accuracy is 85.31%

Using feature(s) {} accuracy is 69.93%
(Warning, Accuracy has decreased! Continuing search in case of local maxima)

Finished search!! The best feature subset is {2}, which has an accuracy of 85.31%
```

```
Welcome to Edwin Leon and Josh McIntyre's Feature Selection Algorithm
Please enter the total number of features
6
Type the number of the algorithm you want to run
1. Forward Selection
2. Backward Elimination
2
Using all features {1, 2, 3, 4, 5, 6} and "random" evaluation, we get an accuracy of 47.78%
Beginning search.
Using feature(s) {2, 3, 4, 5, 6} accuracy is 71.35%
Using feature(s) {1, 3, 4, 5, 6} accuracy is 54.44%
Using feature(s) {1, 2, 4, 5, 6} accuracy is 53.85%
Using feature(s) {1, 2, 3, 5, 6} accuracy is 55.98%
Using feature(s) {1, 2, 3, 4, 6} accuracy is 96.8%
Using feature(s) {1, 2, 3, 4, 5} accuracy is 5.74%

Feature set {1, 2, 3, 4, 6} was best, accuracy is 96.8%

Using feature(s) {2, 3, 4, 6} accuracy is 97.4%
Using feature(s) {1, 3, 4, 6} accuracy is 74.51%
Using feature(s) {1, 2, 4, 6} accuracy is 8.66%
Using feature(s) {1, 2, 3, 6} accuracy is 26.75%
Using feature(s) {1, 2, 3, 4} accuracy is 37.77%

Feature set {2, 3, 4, 6} was best, accuracy is 97.4%

Using feature(s) {3, 4, 6} accuracy is 6.4%
Using feature(s) {2, 4, 6} accuracy is 39.2%
Using feature(s) {2, 3, 6} accuracy is 3.33%
Using feature(s) {2, 3, 4} accuracy is 68.73%

Feature set {2, 3, 4} was best, accuracy is 68.73%

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

Using feature(s) {3, 4} accuracy is 27.96%
Using feature(s) {2, 4} accuracy is 67.02%
Using feature(s) {2, 3} accuracy is 7.29%

Feature set {2, 4} was best, accuracy is 67.02%

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

Using feature(s) {4} accuracy is 60.2%
Using feature(s) {2} accuracy is 47.96%

Feature set {4} was best, accuracy is 60.2%

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

Using feature(s) {} accuracy is 82.79%
Finished search!! The best feature subset is {2, 3, 4, 6}, which has an accuracy of 97.4%
```