WriteUp

In [ ]:
```
##imports
from sklearn.tree import DecisionTreeClassifier
import multiclass
import util
import matplotlib.pyplot as plt
from datasets import *
import linear
import runClassifier
import gd
import numpy as np
```

WU1

(A) Train depth 3 decision trees on the WineDataSmall task. What words are most indicative of being Sauvignon-Blanc? Which words are most indicative of not being Sauvignon-Blanc? What about Pinot-Noir (label==2)?

OAA

In [ ]:
```
##OAA Sauvignon-Blanc
h = multiclass.OAA(5, lambda: DecisionTreeClassifier(max_depth=3))
h.train(WineDataSmall.X, WineDataSmall.Y)
util.showTree(h.f[0], WineDataSmall.words)
```

```
training classifier for 0 versus rest
training classifier for 1 versus rest
training classifier for 2 versus rest
training classifier for 3 versus rest
training classifier for 4 versus rest
gives?
-N-> both?
|    -N-> body?
|    |    -N-> class 0  (356 for class 0, 10 for class 1)
|    |    -Y-> class 1  (0 for class 0, 4 for class 1)
|    -Y-> dry?
|    |    -N-> class 1  (1 for class 0, 15 for class 1)
|    |    -Y-> class 0  (2 for class 0, 0 for class 1)
-Y-> two?
|    -N-> 1?
|    |    -N-> class 1  (4 for class 0, 12 for class 1)
|    |    -Y-> class 0  (11 for class 0, 5 for class 1)
|    -Y-> purity?
|    |    -N-> class 1  (0 for class 0, 14 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
```

In [ ]:
```
##OAA Pinot-Noir
util.showTree(h.f[2], WineDataSmall.words)
```

```
as?
-N-> finish?
|    -N-> medium?
|    |    -N-> class 0  (225 for class 0, 58 for class 1)
|    |    -Y-> class 1  (0 for class 0, 4 for class 1)
|    -Y-> pomegranate?
|    |    -N-> class 1  (0 for class 0, 12 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
-Y-> along?
|    -N-> vibrant?
|    |    -N-> class 1  (36 for class 0, 68 for class 1)
|    |    -Y-> class 0  (8 for class 0, 0 for class 1)
|    -Y-> sonoma?
|    |    -N-> class 0  (21 for class 0, 0 for class 1)
|    |    -Y-> class 1  (0 for class 0, 2 for class 1)
```

In [ ]:

```python
#Sauvignong-Blanc AVA
h = multiclass.AVA(5, lambda: DecisionTreeClassifier(max_depth=3))
h.train(WineDataSmall.X, WineDataSmall.Y)
util.showTree(h.f[1][0], WineDataSmall.words)
print("----------------------------")
util.showTree(h.f[2][0], WineDataSmall.words)
print("----------------------------")
util.showTree(h.f[3][0], WineDataSmall.words)
print("----------------------------")
util.showTree(h.f[4][0], WineDataSmall.words)
```

```
training classifier for 1 versus 0
training classifier for 2 versus 0
training classifier for 2 versus 1
training classifier for 3 versus 0
training classifier for 3 versus 1
training classifier for 3 versus 2
training classifier for 4 versus 0
training classifier for 4 versus 1
training classifier for 4 versus 2
training classifier for 4 versus 3
gives?
-N-> both?
|    -N-> aperitif?
|    |    -N-> class 0  (187 for class 0, 9 for class 1)
|    |    -Y-> class 1  (0 for class 0, 5 for class 1)
|    -Y-> class 1       (0 for class 0, 15 for class 1)
-Y-> class 1    (0 for class 0, 31 for class 1)
---------------------------
boysenberry?
-N-> both?
|    -N-> not?
|    |    -N-> class 0  (141 for class 0, 9 for class 1)
|    |    -Y-> class 1  (0 for class 0, 8 for class 1)
|    -Y-> crimson?
|    |    -N-> class 1  (0 for class 0, 13 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
-Y-> allspice?
|    -N-> class 1       (0 for class 0, 30 for class 1)
|    -Y-> class 0       (2 for class 0, 0 for class 1)
---------------------------
freshly?
-N-> elegant?
|    -N-> coast?
|    |    -N-> class 1  (4 for class 0, 56 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> been?
|    |    -N-> class 1  (1 for class 0, 4 for class 1)
|    |    -Y-> class 0  (4 for class 0, 0 for class 1)
-Y-> class 0    (5 for class 0, 0 for class 1)
---------------------------
across?
-N-> barrels?
|    -N-> length?
|    |    -N-> class 1  (11 for class 0, 56 for class 1)
|    |    -Y-> class 0  (3 for class 0, 0 for class 1)
|    -Y-> class 0       (4 for class 0, 0 for class 1)
-Y-> acidity?
|    -N-> class 0       (10 for class 0, 0 for class 1)
|    -Y-> appealing?
|    |    -N-> class 1  (0 for class 0, 4 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
```

In [ ]:
```
#Pinot-Noir AVA
h = multiclass.AVA(5, lambda: DecisionTreeClassifier(max_depth=3))
h.train(WineDataSmall.X, WineDataSmall.Y)
util.showTree(h.f[2][0], WineDataSmall.words)
```

```
print("----------------------------")
util.showTree(h.f[2][1], WineDataSmall.words)
```

```
training classifier for 1 versus 0
training classifier for 2 versus 0
training classifier for 2 versus 1
training classifier for 3 versus 0
training classifier for 3 versus 1
training classifier for 3 versus 2
training classifier for 4 versus 0
training classifier for 4 versus 1
training classifier for 4 versus 2
training classifier for 4 versus 3
boysenberry?
-N-> both?
|    -N-> not?
|    |    -N-> class 0  (141 for class 0, 9 for class 1)
|    |    -Y-> class 1  (0 for class 0, 8 for class 1)
|    -Y-> flavours?
|    |    -N-> class 1  (0 for class 0, 13 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
-Y-> allspice?
|    -N-> class 1       (0 for class 0, 30 for class 1)
|    -Y-> class 0       (2 for class 0, 0 for class 1)
----------------------------
along?
-N-> aromas?
|    -N-> flavorful?
|    |    -N-> class 1  (92 for class 0, 129 for class 1)
|    |    -Y-> class 0  (11 for class 0, 0 for class 1)
|    -Y-> 2011?
|    |    -N-> class 0  (22 for class 0, 0 for class 1)
|    |    -Y-> class 0  (15 for class 0, 11 for class 1)
-Y-> seafood?
|    -N-> days?
|    |    -N-> class 1  (1 for class 0, 47 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> class 0       (2 for class 0, 0 for class 1)
```

(A)

OAA:

Sauvinon Blanc:

Looking at the tree, we find that word citrus is root, the most differentiating word. When branching 'Y' on Citrus, we find that 31 out of 47 wine are classified as Sauvinon-Blanc (65.95% classified as sauvinon-blanc). When branching 'N' on Citrus, we find that only 29 of 388 are classified as Sauvinon-Blanc (7.22% classified as sauvinon blanc).

Then, looking at inner branch given citrus, we find that branching 'Y' on grapefruit results in 14 out of 15 beeing Sauvinon-blanc. Thus most indicative words of beeing Sauvinon-Blanc is 'citrus' and grapefruit'

Word most undicative of be sauvinon blanc is gooseberry where only 10 of 366(2.73%) were classified as sauvinon-blanc. There less indicative words in terms of rate such as 'or' and

'extremly' (0% sauvinon-blanc) but they only classified 2 and 1 wines respectively and there's no guarantee result will be same on larger sample yet.

Pinot Noir:
Doing similar analysis as above, word most indicative of beeing Pinot Noir was Cherry(51.85% vs 24.6% when present and not present) and raspberries when word cherry is not present (12/13 classified as pinot-noir when word raspberries is Y and cherry is N)

word most un-indicative of beeing Pinot-Noir was verdot where on 'Y' classified 0% as pinot-noir.

AVA:
Sauvinon Blanc:

Too find the most indicative word, looking at trees that are classifying sauvinon-Blanc vs some other class, we find the word that is most likely to classify as sauvinon blanc by looking at rate of classifcation based on the presence of the word.

Word cirtus was most indicative of Sauvinon Blanc classifying 100% of the wine with the word as sauvinon-blanc Thai was least indicative of Sauvinon Blanc calssifiying 0% of wines with the word as suavinon-blanc

Pinot-Noir:

Cassis was most indicative of Pinot-Noir classifying 92% of wines with word present as Pinot-Noir. duck was least indicative of Pino-Noir classifying 0% as Pinot-Noir

(B)

Train depth 3 decision trees on the full WineData task (with 20 labels). What accuracy do you get? How long does this take (in seconds)? One of my least favorite wines is Viognier -- what words are indicative of this?

In [ ]:
```
#OAA
oaa = multiclass.OAA(20, lambda: DecisionTreeClassifier(max_depth=3))

start = time.time()
oaa.train(WineData.X, WineData.Y)
end = time.time()
predict =oaa.predictAll(WineData.Xte)

print("Time:" + str(end - start))
print("Accuracy:" + str(mean(predict == WineData.Yte) * 100) +"%\n")
util.showTree(oaa.f[17], WineData.words)
```

```
training classifier for 0 versus rest
training classifier for 1 versus rest
training classifier for 2 versus rest
training classifier for 3 versus rest
training classifier for 4 versus rest
training classifier for 5 versus rest
training classifier for 6 versus rest
training classifier for 7 versus rest
training classifier for 8 versus rest
training classifier for 9 versus rest
training classifier for 10 versus rest
training classifier for 11 versus rest
training classifier for 12 versus rest
training classifier for 13 versus rest
training classifier for 14 versus rest
training classifier for 15 versus rest
training classifier for 16 versus rest
training classifier for 17 versus rest
training classifier for 17 versus rest
training classifier for 18 versus rest
training classifier for 19 versus rest
Time:0.15854930877685547
Accuracy:36.82745825602968%

to?
-N-> up?
|    -N-> they?
|    |    -N-> class 0  (1036 for class 0, 1 for class 1)
|    |    -Y-> class 0  (6 for class 0, 1 for class 1)
|    -Y-> supported?
|    |    -N-> class 0  (13 for class 0, 1 for class 1)
|    |    -Y-> class 1  (0 for class 0, 1 for class 1)
-Y-> shellfish?
|    -N-> lead?
|    |    -N-> class 0  (14 for class 0, 0 for class 1)
|    |    -Y-> class 1  (0 for class 0, 1 for class 1)
|    -Y-> class 1       (0 for class 0, 3 for class 1)
```

In [ ]:

```
#AVA
ava = multiclass.AVA(20, lambda: DecisionTreeClassifier(max_depth=3))

start = time.time()
ava.train(WineData.X, WineData.Y)
end = time.time()
predict =ava.predictAll(WineData.Xte)

print("Time:" + str(end - start))
print("Accuracy:" + str(mean(predict == WineData.Yte) * 100) +"%\n")

for i in range(17):
    util.showTree(ava.f[17][i], WineData.words)
    print("------------------------------------")
```

```
training classifier for 1 versus 0
training classifier for 2 versus 0
training classifier for 2 versus 1
training classifier for 3 versus 0
training classifier for 3 versus 1
training classifier for 3 versus 2
training classifier for 4 versus 0
training classifier for 4 versus 1
training classifier for 4 versus 2
training classifier for 4 versus 3
training classifier for 5 versus 0
training classifier for 5 versus 1
training classifier for 5 versus 2
training classifier for 5 versus 3
training classifier for 5 versus 4
training classifier for 6 versus 0
training classifier for 6 versus 1
training classifier for 6 versus 2
training classifier for 6 versus 3
training classifier for 6 versus 4
training classifier for 6 versus 5
training classifier for 7 versus 0
training classifier for 7 versus 1
training classifier for 7 versus 2
training classifier for 7 versus 3
training classifier for 7 versus 4
training classifier for 7 versus 5
training classifier for 7 versus 6
training classifier for 8 versus 0
training classifier for 8 versus 1
training classifier for 8 versus 2
training classifier for 8 versus 3
training classifier for 8 versus 4
training classifier for 8 versus 5
training classifier for 8 versus 6
training classifier for 8 versus 7
training classifier for 9 versus 0
training classifier for 9 versus 1
training classifier for 9 versus 2
training classifier for 9 versus 3
training classifier for 9 versus 4
training classifier for 9 versus 5
training classifier for 9 versus 6
training classifier for 9 versus 7
training classifier for 9 versus 8
training classifier for 10 versus 0
training classifier for 10 versus 1
training classifier for 10 versus 2
training classifier for 10 versus 3
training classifier for 10 versus 4
training classifier for 10 versus 5
training classifier for 10 versus 6
training classifier for 10 versus 7
training classifier for 10 versus 8
training classifier for 10 versus 9
training classifier for 11 versus 0
```

```
training classifier for 11 versus 1
training classifier for 11 versus 2
training classifier for 11 versus 3
training classifier for 11 versus 4
training classifier for 11 versus 5
training classifier for 11 versus 6
training classifier for 11 versus 7
training classifier for 11 versus 8
training classifier for 11 versus 9
training classifier for 11 versus 10
training classifier for 12 versus 0
training classifier for 12 versus 1
training classifier for 12 versus 2
training classifier for 12 versus 3
training classifier for 12 versus 4
training classifier for 12 versus 5
training classifier for 12 versus 6
training classifier for 12 versus 7
training classifier for 12 versus 8
training classifier for 12 versus 9
training classifier for 12 versus 10
training classifier for 12 versus 11
training classifier for 13 versus 0
training classifier for 13 versus 1
training classifier for 13 versus 2
training classifier for 13 versus 3
training classifier for 13 versus 4
training classifier for 13 versus 5
training classifier for 13 versus 6
training classifier for 13 versus 7
training classifier for 13 versus 8
training classifier for 13 versus 9
training classifier for 13 versus 10
training classifier for 13 versus 11
training classifier for 13 versus 12
training classifier for 14 versus 0
training classifier for 14 versus 1
training classifier for 14 versus 2
training classifier for 14 versus 3
training classifier for 14 versus 4
training classifier for 14 versus 5
training classifier for 14 versus 6
training classifier for 14 versus 7
training classifier for 14 versus 8
training classifier for 14 versus 9
training classifier for 14 versus 10
training classifier for 14 versus 11
training classifier for 14 versus 12
training classifier for 14 versus 13
training classifier for 15 versus 0
training classifier for 15 versus 1
training classifier for 15 versus 2
training classifier for 15 versus 3
training classifier for 15 versus 4
training classifier for 15 versus 5
training classifier for 15 versus 6
```

```
training classifier for 15 versus 7
training classifier for 15 versus 8
training classifier for 15 versus 9
training classifier for 15 versus 10
training classifier for 15 versus 11
training classifier for 15 versus 12
training classifier for 15 versus 13
training classifier for 15 versus 14
training classifier for 16 versus 0
training classifier for 16 versus 1
training classifier for 16 versus 2
training classifier for 16 versus 3
training classifier for 16 versus 4
training classifier for 16 versus 5
training classifier for 16 versus 6
training classifier for 16 versus 7
training classifier for 16 versus 8
training classifier for 16 versus 9
training classifier for 16 versus 10
training classifier for 16 versus 11
training classifier for 16 versus 12
training classifier for 16 versus 13
training classifier for 16 versus 14
training classifier for 16 versus 15
training classifier for 17 versus 0
training classifier for 17 versus 1
training classifier for 17 versus 2
training classifier for 17 versus 3
training classifier for 17 versus 4
training classifier for 17 versus 5
training classifier for 17 versus 6
training classifier for 17 versus 7
training classifier for 17 versus 8
training classifier for 17 versus 9
training classifier for 17 versus 10
training classifier for 17 versus 11
training classifier for 17 versus 12
training classifier for 17 versus 13
training classifier for 17 versus 14
training classifier for 17 versus 15
training classifier for 17 versus 16
training classifier for 18 versus 0
training classifier for 18 versus 1
training classifier for 18 versus 2
training classifier for 18 versus 3
training classifier for 18 versus 4
training classifier for 18 versus 5
training classifier for 18 versus 6
training classifier for 18 versus 7
training classifier for 18 versus 8
training classifier for 18 versus 9
training classifier for 18 versus 10
training classifier for 18 versus 11
training classifier for 18 versus 12
training classifier for 18 versus 13
training classifier for 18 versus 14
```

```
training classifier for 18 versus 15
training classifier for 18 versus 16
training classifier for 18 versus 17
training classifier for 19 versus 0
training classifier for 19 versus 1
training classifier for 19 versus 2
training classifier for 19 versus 3
training classifier for 19 versus 4
training classifier for 19 versus 5
training classifier for 19 versus 6
training classifier for 19 versus 7
training classifier for 19 versus 8
training classifier for 19 versus 9
training classifier for 19 versus 10
training classifier for 19 versus 11
training classifier for 19 versus 12
training classifier for 19 versus 13
training classifier for 19 versus 14
training classifier for 19 versus 15
training classifier for 19 versus 16
training classifier for 19 versus 17
training classifier for 19 versus 18
Time:0.23581838607788086
Accuracy:27.17996289424861%

add?
-N-> just?
|    -N-> crimson?
|    |    -N-> class 1  (2 for class 0, 59 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> class 0       (1 for class 0, 0 for class 1)
-Y-> end?
|    -N-> class 0       (4 for class 0, 0 for class 1)
|    -Y-> class 1       (0 for class 0, 1 for class 1)
-----------------------------------
to?
-N-> apples?
|    -N-> aged?
|    |    -N-> class 1  (0 for class 0, 187 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> class 0       (3 for class 0, 0 for class 1)
-Y-> class 0    (4 for class 0, 0 for class 1)
-----------------------------------
to?
-N-> apples?
|    -N-> aged?
|    |    -N-> class 1  (0 for class 0, 144 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> class 0       (3 for class 0, 0 for class 1)
-Y-> class 0    (4 for class 0, 0 for class 1)
-----------------------------------
to?
-N-> up?
|    -N-> powerful?
|    |    -N-> class 1  (1 for class 0, 15 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
```

```
|    -Y-> class 0       (2 for class 0, 0 for class 1)
-Y-> class 0    (4 for class 0, 0 for class 1)
-----------------------------------
to?
-N-> overtones?
|    -N-> berries?
|    |    -N-> class 1  (1 for class 0, 29 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> class 0       (2 for class 0, 0 for class 1)
-Y-> class 0    (4 for class 0, 0 for class 1)
-----------------------------------
shellfish?
-N-> green?
|    -N-> peaches?
|    |    -N-> class 1  (3 for class 0, 147 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> class 0       (1 for class 0, 0 for class 1)
-Y-> class 0    (3 for class 0, 0 for class 1)
-----------------------------------
to?
-N-> it?
|    -N-> aroma?
|    |    -N-> class 1  (1 for class 0, 61 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> class 0       (2 for class 0, 0 for class 1)
-Y-> class 0    (4 for class 0, 0 for class 1)
-----------------------------------
add?
-N-> aged?
|    -N-> bottling?
|    |    -N-> class 1  (0 for class 0, 57 for class 1)
|    |    -Y-> class 0  (2 for class 0, 0 for class 1)
|    -Y-> class 0       (2 for class 0, 0 for class 1)
-Y-> class 0    (4 for class 0, 0 for class 1)
-----------------------------------
to?
-N-> apples?
|    -N-> across?
|    |    -N-> class 1  (0 for class 0, 47 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> class 0       (3 for class 0, 0 for class 1)
-Y-> class 0    (4 for class 0, 0 for class 1)
-----------------------------------
to?
-N-> apples?
|    -N-> seamlessly?
|    |    -N-> class 1  (0 for class 0, 49 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> class 0       (3 for class 0, 0 for class 1)
-Y-> class 0    (4 for class 0, 0 for class 1)
-----------------------------------
to?
-N-> apples?
|    -N-> across?
|    |    -N-> class 1  (0 for class 0, 48 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
```

```
|    -Y-> class 0       (3 for class 0, 0 for class 1)
-Y-> class 0    (4 for class 0, 0 for class 1)
-----------------------------------
(?
-N-> apples?
|    -N-> aged?
|    |    -N-> class 1  (0 for class 0, 31 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> class 0       (3 for class 0, 0 for class 1)
-Y-> class 0    (4 for class 0, 0 for class 1)
-----------------------------------
shellfish?
-N-> bottling?
|    -N-> this?
|    |    -N-> class 1  (2 for class 0, 34 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> favorite?
|    |    -N-> class 0  (2 for class 0, 0 for class 1)
|    |    -Y-> class 1  (0 for class 0, 1 for class 1)
-Y-> class 0    (3 for class 0, 0 for class 1)
-----------------------------------
apples?
-N-> to?
|    -N-> across?
|    |    -N-> class 1  (0 for class 0, 45 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> class 0       (3 for class 0, 0 for class 1)
-Y-> class 0    (4 for class 0, 0 for class 1)
-----------------------------------
to?
-N-> apples?
|    -N-> across?
|    |    -N-> class 1  (0 for class 0, 35 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> class 0       (3 for class 0, 0 for class 1)
-Y-> class 0    (4 for class 0, 0 for class 1)
-----------------------------------
love?
-N-> it?
|    -N-> overtones?
|    |    -N-> class 1  (1 for class 0, 31 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> friendly?
|    |    -N-> class 0  (3 for class 0, 0 for class 1)
|    |    -Y-> class 1  (0 for class 0, 1 for class 1)
-Y-> class 0    (3 for class 0, 0 for class 1)
-----------------------------------
apples?
-N-> to?
|    -N-> overtones?
|    |    -N-> class 1  (0 for class 0, 16 for class 1)
|    |    -Y-> class 0  (1 for class 0, 0 for class 1)
|    -Y-> class 0       (3 for class 0, 0 for class 1)
-Y-> class 0    (4 for class 0, 0 for class 1)
-----------------------------------
```

(B)
OAA:
Time:0.1535322666168213 seconds
Accuracy:36.641929499072354%
Words most indicative of Viognier was Peach and Milk when comibined classified as Viognier 100% of the time.

AVA
Time:0.23581671714782715
Accuracy:26.90166975881262%

In AVA most words didn't indicate Viognier. Instead, absense of certain words were strong indicator of Viognier such as fragrant, tannins, pear, apple, jasmine, edge etc...

(C)

Compare the accuracy in (B) using zero-one predictions versus using confidence. How much difference does it make?

In [ ]:
```
#OAA#
predict = oaa.predictAll(WineData.Xte, useZeroOne=True)
print("OAA Accuracy:" + str(mean(predict == WineData.Yte) * 100) +"%\n")

#AVA#
predict = ava.predictAll(WineData.Xte, useZeroOne=True)
print("AVA Accuracy:" + str(mean(predict == WineData.Yte) * 100) +"%\n")
```

```
OAA Accuracy:24.58256029684601%

AVA Accuracy:26.808905380333954%
```

OAA Accuracy using ZeroOne is 24.860853432282003% which is about 11.78%p worse than 36.641929499072354%
AVA accuracy using ZeroOne is 26.34508348794063% which is 0.56%p worse than 26.90166975881262%

WU2

Show the test accuracy you get with a balanced tree on the WineData using a DecisionTreeClassifier with max depth 3.

In [ ]:
```
t = multiclass.makeBalancedTree(range(20))
h = multiclass.MCTree(t, lambda: DecisionTreeClassifier(max_depth=3))
h.train(WineData.X, WineData.Y)
p = h.predictAll(WineData.Xte)
```

```
print ("Accuracy:" + str(mean(p == WineData.Yte) * 100) + "%")
```

```
training classifier for [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] versus [10, 11, 12, 13, 14, 1
5, 16, 17, 18, 19]
training classifier for [0, 1, 2, 3, 4] versus [5, 6, 7, 8, 9]
training classifier for [0, 1] versus [2, 3, 4]
training classifier for [0] versus [1]
training classifier for [2] versus [3, 4]
training classifier for [3] versus [4]
training classifier for [5, 6] versus [7, 8, 9]
training classifier for [5] versus [6]
training classifier for [7] versus [8, 9]
training classifier for [8] versus [9]
training classifier for [10, 11, 12, 13, 14] versus [15, 16, 17, 18, 19]
training classifier for [10, 11] versus [12, 13, 14]
training classifier for [10] versus [11]
training classifier for [12] versus [13, 14]
training classifier for [13] versus [14]
training classifier for [15, 16] versus [17, 18, 19]
training classifier for [15] versus [16]
training classifier for [17] versus [18, 19]
training classifier for [18] versus [19]
Accuracy:13.543599257884972%
```

Accuracy on balanced tree on winedata using DecisionTreeClassifier with max depth 3 is about 13.54%

WU3

What is the impact of the step size on convergence? Find values of the step size where the algorithm diverges and converges.

Step size impact convergence as its the amount of step we descent towards optima. Bigger the step size, quicker we converge, however, if step size is too big, may result in never converging due to overshooting past the minima.

Using 100 iteration and changes of 0.5 to step sizes, At negative values of step size, algorithm does not converge, Algorithm converges startin around 0.5 step size and start to diverge at around 5.5 step size

In [ ]:
```
for i in linspace(-1,8,19):
    x, trajectory = gd.gd(lambda x: x**2, lambda x: 2*x, 10, 100, i)
    print("step size " + str(i) +": "  +str(x))
```

```
step size -1.0: 137989559296995.62
step size -0.5: 153615668.45838782
step size 0.0: 10.0
step size 0.5: 0.0
step size 1.0: 0.0
step size 1.5: 0.0
step size 2.0: 0.0
step size 2.5: 0.0
step size 3.0: 0.0
step size 3.5: 0.0
step size 4.0: 0.0
step size 4.5: 0.0
step size 5.0: 0.0
step size 5.5: 1.0525679427688571e-22
step size 6.0: 9.107243088758273e-11
step size 6.5: 0.07329778064924164
step size 7.0: 1192445.5000873771
step size 7.5: 1487861268643.2266
step size 8.0: 2.942427967911726e+17
```

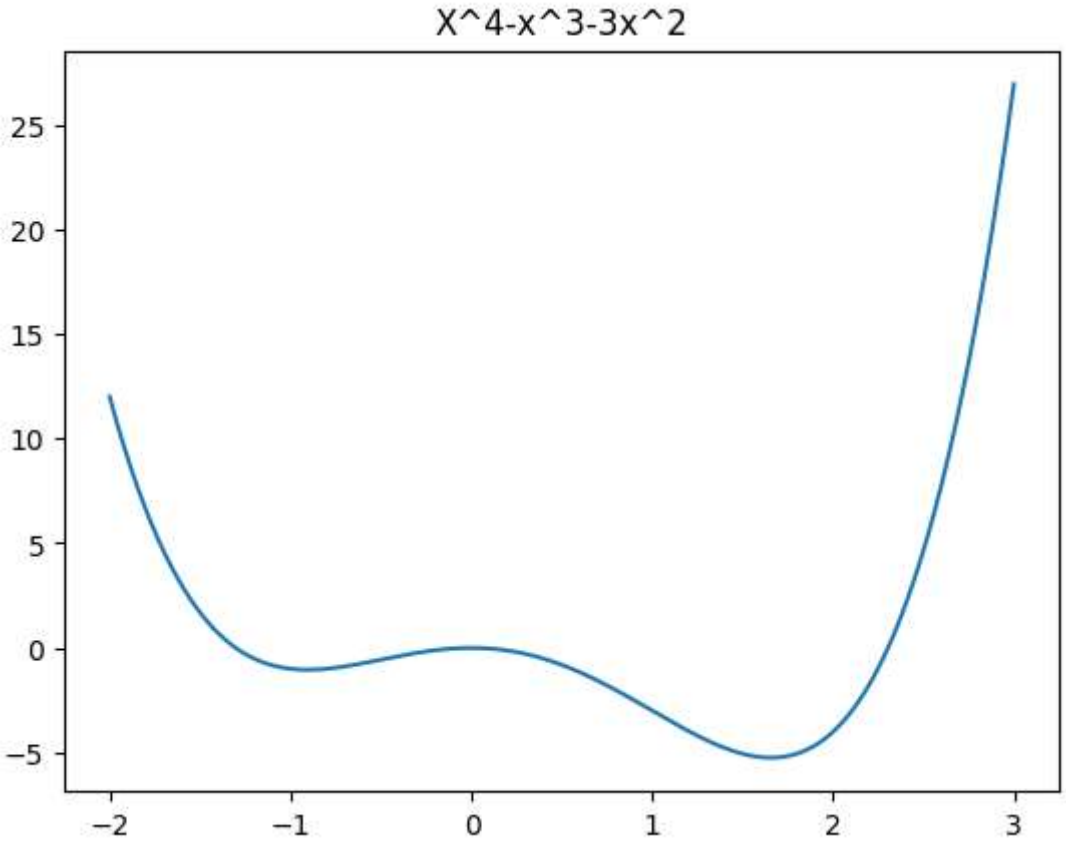
WU4

Come up with a non-convex univariate optimization problem. Plot the function you're trying to minimize and show two runs of gd, one where it gets caught in a local minimum and one where it manages to make it to a global minimum. (Use different starting points to accomplish this.)

f = x^4-x^3-3x^2 is a univariate non-covex function with global minimum at (1.656, -5.248) and a local minimum (-0.906,-1.045)

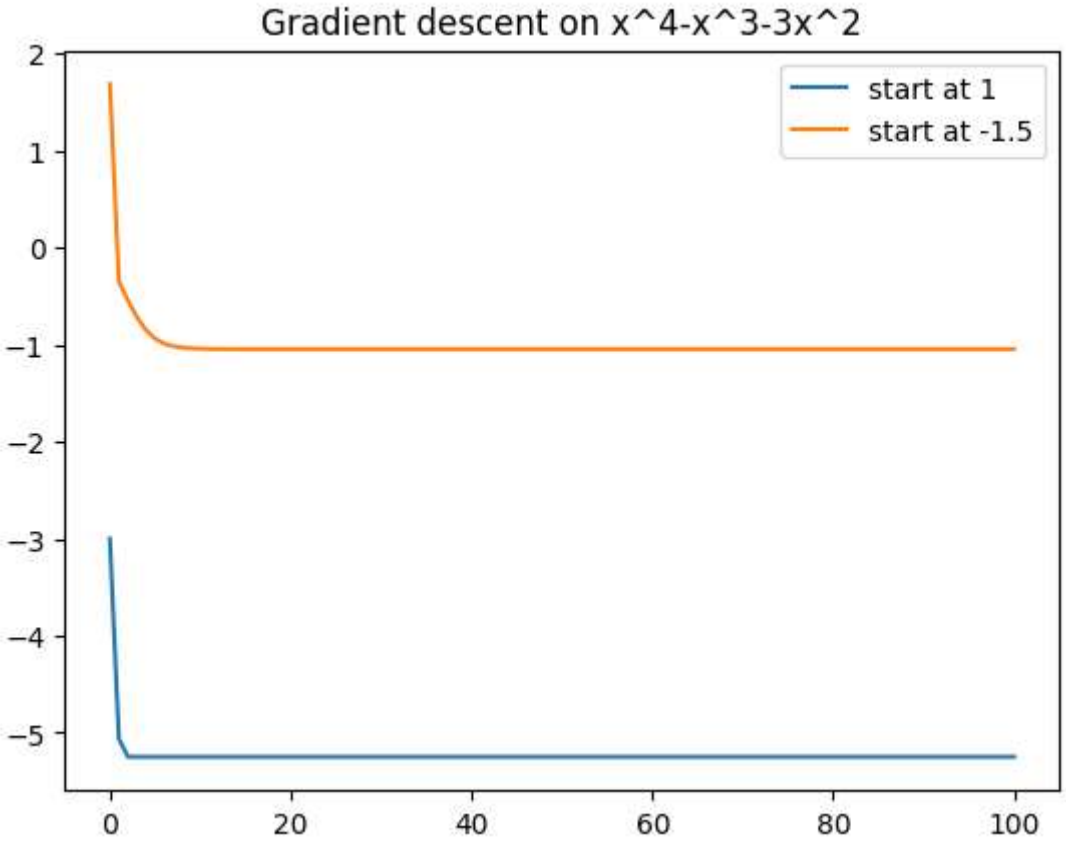
In [ ]:
```
#graph of f.
x = np.arange(-2,3,0.001)
f = x**4 - x**3 - 3*x**2
plt.plot(x,f)
plt.title("X^4-x^3-3x^2")
```

Out[ ]:
```
Text(0.5, 1.0, 'X^4-x^3-3x^2')
```



Finding gradient: d/dx (fx) = 4x^3 - 3x^2 - 6x Using 100 iteration and step size of 0.1

In [ ]:
```
#gradient descent
x, trajectory = gd.gd(lambda x: x**4 - x**3 - 3*x**2, lambda x: 4*x**3-3*x**2-6*x,
plt.plot(trajectory, label = "start at 1")
x2, trajectory2 = gd.gd(lambda x: x**4 - x**3 - 3*x**2, lambda x: 4*x**3-3*x**2-6*x
plt.plot(trajectory2, label = "start at -1.5")
plt.legend()
plt.title("Gradient descent on x^4-x^3-3x^2")
plt.show()
```



In [ ]:
```
#printing out x as result of gd.
print(x)
print(x2)
```

```
1.6558688457449502
-0.9058688084239377
```

We see that when starting at 1, gd was able to find global minimum But when we started at -1.5, gd got stuck in local minimum

WU5

For each of the loss functions, train a model on the binary version of the wine data (called WineDataBinary) and evaluate it on the test data. You should use lambda=1 in all cases. Which works best? For that best model, look at the learned weights. Find the words corresponding to the weights with the greatest positive value and those with the greatest negative value. Hint: look at WineDataBinary.words to get the id-to-word mapping. List the top 5 positive and top 5 negative and explain.

In [ ]:
```
print("square loss")
sq = linear.LinearClassifier({'lossFunction': linear.SquaredLoss(), 'lambda': 1, 'n
runClassifier.trainTestSet(sq,WineDataBinary)
print("hinge loss")
hinge = linear.LinearClassifier({'lossFunction': linear.HingeLoss(), 'lambda': 1, '
runClassifier.trainTestSet(hinge,WineDataBinary)
print("logistic loss")
```

```
log = linear.LinearClassifier({'lossFunction': linear.LogisticLoss(), 'lambda': 1,
runClassifier.trainTestSet(log,WineDataBinary)

#retreiving words
words = WineDataBinary.words


#mapping weight to word as list of tuple
result = list(zip(log.getRepresentation(),words))
#sort by weight
result = sorted(result)

print("five most negative words")
neg = result[:5]
print(neg)


print("five most positive result")
pos = result[-5:]
print(pos)
```

```
square loss
Training accuracy 0.242915, test accuracy 0.313653
hinge loss
Training accuracy 0.753036, test accuracy 0.686347
logistic loss
Training accuracy 0.995951, test accuracy 0.97417
five most negative words
[(-1.1695212164040438, '2011'), (-0.7653093906427083, '10'), (-0.6835931677893792,
'%'), (-0.6295907281434366, 'as'), (-0.5321916724675304, 'between')]
five most positive result
[(0.6064232619016866, 'approachable'), (0.6891990079029966, 'aromas'), (0.7108905521
536923, 'both'), (0.7701247691557759, 'boysenberry'), (0.8832897531176551, 'gives')]
```

Logistic loss had best result with training accuracy of 95.59% and test accuracy of 94.41%

Thus, getting the learned weigts from logistic loss model, mapping to the words from WineDataBinary, we have Top five most positive:

1. 'gives' with weight 0.8832897531176551
2. 'boysenberry' with weight 0.7701247691557759
3. 'both' with weight 0.7108905521536923
4. 'aromas' with weight 0.6891990079029966
5. 'approachable' with weight 0.6064232619016866

Top 5 most negative:
the positive words are mostly words that contribute to most to predicting positive class.

1. '2011' with weight -1.1695212164040438
2. '10' with weight -0.7653093906427083
3. '%' with weight -0.6835931677893792
4. 'as' with weight -0.6295907281434366

5. 'between' with weight -0.5321916724675304

negative are the words that contribute most to predicting negative class.