Adaboost from scratch

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from random import sample
import random
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix
from sklearn import tree
from math import log,exp
from sklearn.datasets import load_iris
```

```
pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
```

```
iris = load_iris()
```

```
iris
```

```
{'data': array([[5.1, 3.5, 1.4, 0.2],
        [4.9, 3. , 1.4, 0.2],
        [4.7, 3.2, 1.3, 0.2],
        [4.6, 3.1, 1.5, 0.2],
        [5. , 3.6, 1.4, 0.2],
        [5.4, 3.9, 1.7, 0.4],
        [4.6, 3.4, 1.4, 0.3],
        [5. , 3.4, 1.5, 0.2],
        [4.4, 2.9, 1.4, 0.2],
        [4.9, 3.1, 1.5, 0.1],
        [5.4, 3.7, 1.5, 0.2],
        [4.8, 3.4, 1.6, 0.2],
        [4.8, 3. , 1.4, 0.1],
        [4.3, 3. , 1.1, 0.1],
        [5.8, 4. , 1.2, 0.2],
        [5.7, 4.4, 1.5, 0.4],
        [5.4, 3.9, 1.3, 0.4],
        [5.1, 3.5, 1.4, 0.3],
        [5.7, 3.8, 1.7, 0.3],
        [5.1, 3.8, 1.5, 0.3],
        [5.4, 3.4, 1.7, 0.2],
        [5.1, 3.7, 1.5, 0.4],
        [4.6, 3.6, 1. , 0.2],
        [5.1, 3.3, 1.7, 0.5],
        [4.8, 3.4, 1.9, 0.2],
        [5. , 3. , 1.6, 0.2],
        [5. , 3.4, 1.6, 0.4],
        [5.2, 3.5, 1.5, 0.2],
        [5.2, 3.4, 1.4, 0.2],
        [4.7, 3.2, 1.6, 0.2],
        [4.8, 3.1, 1.6, 0.2],
        [5.4, 3.4, 1.5, 0.4],
        [5.2, 4.1, 1.5, 0.1],
        [5.5, 4.2, 1.4, 0.2],
        [4.9, 3.1, 1.5, 0.2],
        [5. , 3.2, 1.2, 0.2],
        [5.5, 3.5, 1.3, 0.2],
        [4.9, 3.6, 1.4, 0.1],
        [4.4, 3. , 1.3, 0.2],
        [5.1, 3.4, 1.5, 0.2],
        [5. , 3.5, 1.3, 0.3],
        [4.5, 2.3, 1.3, 0.3],
        [4.4, 3.2, 1.3, 0.2],
        [5. , 3.5, 1.6, 0.6],
        [5.1, 3.8, 1.9, 0.4],
        [4.8, 3. , 1.4, 0.3],
        [5.1, 3.8, 1.6, 0.2],
        [4.6, 3.2, 1.4, 0.2],
        [5.3, 3.7, 1.5, 0.2],
        [5. , 3.3, 1.4, 0.2],
        [7. , 3.2, 4.7, 1.4],
        [6.4, 3.2, 4.5, 1.5],
        [6.9, 3.1, 4.9, 1.5],
        [5.5, 2.3, 4. , 1.3],
        [6.5, 2.8, 4.6, 1.5],
        [5.7, 2.8, 4.5, 1.3],
        [6.3, 3.3, 4.7, 1.6],
        [4.9, 2.4, 3.3, 1. ],
```

```
iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
```

```
iris_df['species'] = iris.target

# Optional: You can map the target numbers to their corresponding species names
iris_df['species'] = iris_df['species'].map({0: 'setosa', 1: 'versicolor', 2: 'virginica'})
```

```
iris_df.head(1)
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	5.1	3.5	1.4	0.2	setosa

```
iris=iris_df
```

```
example = iris[(iris['species'] == 'versicolor') | (iris['species'] == 'virginica')]
```

```
example.head(2)
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
50	7.0	3.2	4.7	1.4	versicolor
51	6.4	3.2	4.5	1.5	versicolor

```
example['Label'] = example['species'].replace(to_replace = ['versicolor','virginica'], value=[1,-1])
```

```
<ipython-input-22-cd8732b3801d>:1: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a future ve
  example['Label'] = example['species'].replace(to_replace = ['versicolor','virginica'], value=[1,-1])
<ipython-input-22-cd8732b3801d>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  example['Label'] = example['species'].replace(to_replace = ['versicolor','virginica'], value=[1,-1])
```

```
example = example.drop('species', axis = 1)
```

```
example['probR1'] = 1/(example.shape[0])
```

```
example.head(5)
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Label	probR1
50	7.0	3.2	4.7	1.4	1	0.01
51	6.4	3.2	4.5	1.5	1	0.01
52	6.9	3.1	4.9	1.5	1	0.01
53	5.5	2.3	4.0	1.3	1	0.01
54	6.5	2.8	4.6	1.5	1	0.01

```
random.seed(10)
example1 = example.sample(len(example), replace = True, weights = example['probR1'])
```

```
example1
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Label	probR1
94	5.6	2.7	4.2	1.3	1	0.01
60	5.0	2.0	3.5	1.0	1	0.01
86	6.7	3.1	4.7	1.5	1	0.01
86	6.7	3.1	4.7	1.5	1	0.01
145	6.7	3.0	5.2	2.3	-1	0.01
...	...	...	...	...	...	...
122	7.7	2.8	6.7	2.0	-1	0.01
141	6.9	3.1	5.1	2.3	-1	0.01
63	6.1	2.9	4.7	1.4	1	0.01
131	7.9	3.8	6.4	2.0	-1	0.01
77	6.7	3.0	5.0	1.7	1	0.01

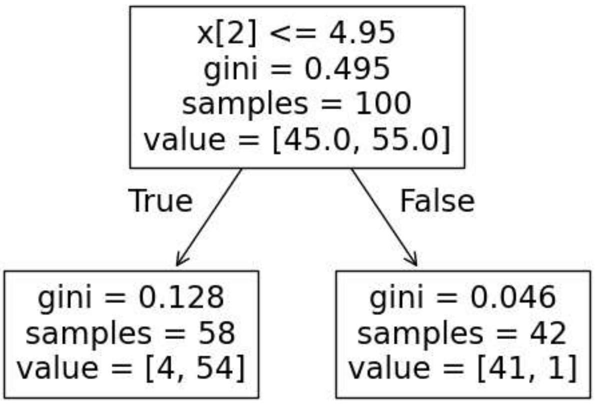
100 rows × 6 columns

```
#X_train and Y_train split
X_train = example1.iloc[0:len(iris),0:4]
y_train = example1.iloc[0:len(iris),4]
```

```
#fitting the DT model with depth one
clf_gini = DecisionTreeClassifier(criterion = "gini", random_state = 100, max_depth=1)
clf = clf_gini.fit(X_train, y_train)
```

```
#plotting tree for round 1 boosting
tree.plot_tree(clf)
```

```
[Text(0.5, 0.75, 'x[2] <= 4.95\ngini = 0.495\nsamples = 100\nvalue = [45.0, 55.0]'),
 Text(0.25, 0.25, 'gini = 0.128\nsamples = 58\nvalue = [4, 54]'),
 Text(0.375, 0.5, 'True  '),
 Text(0.75, 0.25, 'gini = 0.046\nsamples = 42\nvalue = [41, 1]'),
 Text(0.625, 0.5, '  False')]
```




```
#prediction
y_pred = clf_gini.predict(example.iloc[0:len(iris),0:4])
y_pred
```

```
array([ 1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
        1,  1,  1,  1,  1,  1,  1,  1,  1,  1, -1,  1,  1,  1,  1,  1, -1,
        1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1, -1,
       -1, -1, -1, -1, -1,  1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
       -1, -1, -1,  1, -1,  1, -1, -1,  1,  1, -1, -1, -1, -1, -1, -1, -1,
       -1, -1, -1,  1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1])
```

```
#adding a column pred1 after the first round of boosting
example['pred1'] = y_pred
```

```
example
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Label	probR1	pred1
50	7.0	3.2	4.7	1.4	1	0.01	1
51	6.4	3.2	4.5	1.5	1	0.01	1
52	6.9	3.1	4.9	1.5	1	0.01	1
53	5.5	2.3	4.0	1.3	1	0.01	1
54	6.5	2.8	4.6	1.5	1	0.01	1
...	...	...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	-1	0.01	-1
146	6.3	2.5	5.0	1.9	-1	0.01	-1
147	6.5	3.0	5.2	2.0	-1	0.01	-1
148	6.2	3.4	5.4	2.3	-1	0.01	-1
149	5.9	3.0	5.1	1.8	-1	0.01	-1

100 rows × 7 columns

```
#misclassified = 0 if the label and prediction are same
example.loc[example.Label != example.pred1, 'misclassified'] = 1
example.loc[example.Label == example.pred1, 'misclassified'] = 0
```

```
#error calculation
e1 = sum(example['misclassified'] * example['probR1'])
```

```
e1
```

```
0.08
```

```
#calculation of alpha (performance)
alpha1 = 0.5*log((1-e1)/e1)
```

```
#update weight
new_weight = example['probR1']*np.exp(-1*alpha1*example['Label']*example['pred1'])
```

```
#normalized weight
z = sum(new_weight)
normalized_weight = new_weight/sum(new_weight)
example['prob2'] = round(normalized_weight,4)
```

```
example
```

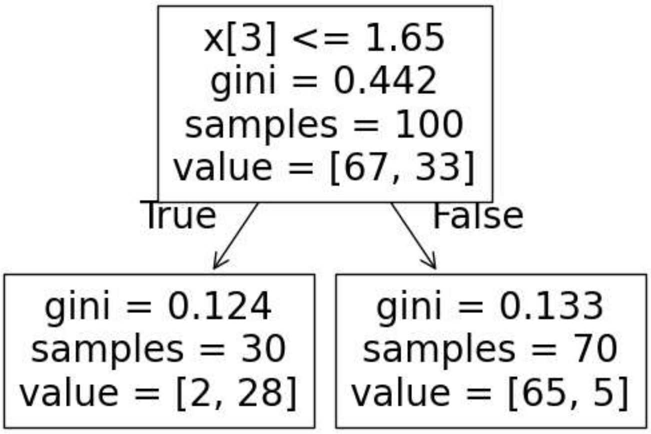
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Label	probR1	pred1	misclassified	prob2
50	7.0	3.2	4.7	1.4	1	0.01	1	0.0	0.0054
51	6.4	3.2	4.5	1.5	1	0.01	1	0.0	0.0054
52	6.9	3.1	4.9	1.5	1	0.01	1	0.0	0.0054
53	5.5	2.3	4.0	1.3	1	0.01	1	0.0	0.0054
54	6.5	2.8	4.6	1.5	1	0.01	1	0.0	0.0054
...	...	...	...	...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	-1	0.01	-1	0.0	0.0054
146	6.3	2.5	5.0	1.9	-1	0.01	-1	0.0	0.0054
147	6.5	3.0	5.2	2.0	-1	0.01	-1	0.0	0.0054
148	6.2	3.4	5.4	2.3	-1	0.01	-1	0.0	0.0054
149	5.9	3.0	5.1	1.8	-1	0.01	-1	0.0	0.0054

100 rows × 9 columns

```
#round 2
random.seed(20)
example2 = example.sample(len(example), replace = True, weights = example['prob2'])
example2 = example2.iloc[:,0:5]
X_train = example2.iloc[0:len(iris),0:4]
y_train = example2.iloc[0:len(iris),4]

clf_gini = DecisionTreeClassifier(criterion = "gini", random_state = 100, max_depth=1)
```

```
clf = clf_gini.fit(X_train, y_train)

y_pred = clf_gini.predict(example.iloc[0:len(iris),0:4])
#adding a column pred2 after the second round of boosting
example['pred2'] = y_pred


#plotting tree for round 2 boosting
tree.plot_tree(clf)
```

```
[Text(0.5, 0.75, 'x[3] <= 1.65\ngini = 0.442\nsamples = 100\nvalue = [67, 33]'),
 Text(0.25, 0.25, 'gini = 0.124\nsamples = 30\nvalue = [2, 28]'),
 Text(0.375, 0.5, 'True  '),
 Text(0.75, 0.25, 'gini = 0.133\nsamples = 70\nvalue = [65, 5]'),
 Text(0.625, 0.5, '  False')]
```




```
example
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Label	probR1	pred1	misclassified	prob2	pred2
50	7.0	3.2	4.7	1.4	1	0.01	1	0.0	0.0054	1
51	6.4	3.2	4.5	1.5	1	0.01	1	0.0	0.0054	1
52	6.9	3.1	4.9	1.5	1	0.01	1	0.0	0.0054	1
53	5.5	2.3	4.0	1.3	1	0.01	1	0.0	0.0054	1
54	6.5	2.8	4.6	1.5	1	0.01	1	0.0	0.0054	1
...	...	...	...	...	...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	-1	0.01	-1	0.0	0.0054	-1
146	6.3	2.5	5.0	1.9	-1	0.01	-1	0.0	0.0054	-1
147	6.5	3.0	5.2	2.0	-1	0.01	-1	0.0	0.0054	-1
148	6.2	3.4	5.4	2.3	-1	0.01	-1	0.0	0.0054	-1
149	5.9	3.0	5.1	1.8	-1	0.01	-1	0.0	0.0054	-1

100 rows × 10 columns

```
#adding a field misclassified2
example.loc[example.Label != example.pred2, 'misclassified2'] = 1
example.loc[example.Label == example.pred2, 'misclassified2'] = 0
# calculation of error
e2 = sum(example['misclassified2'] * example['prob2'])
e2
```

```
0.08950000000000001
```

```
#calculation of alpha
alpha2 = 0.5*log((1-e2)/e2)
alpha2
```

```
1.1598776369434263
```

```
#update weight
new_weight = example['prob2']*np.exp(-1*alpha2*example['Label']*example['pred2'])
z = sum(new_weight)
```

```
normalized_weight = new_weight/sum(new_weight)
example['prob3'] = round(normalized_weight,4)
```

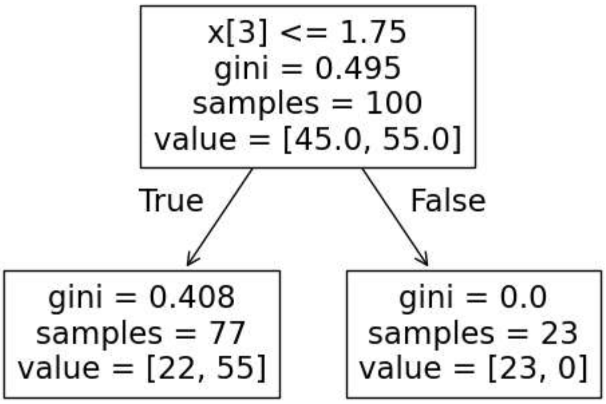
```
example
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Label	probR1	pred1	misclassified	prob2	pred2	misclassified2	prob3
50	7.0	3.2	4.7	1.4	1	0.01	1	0.0	0.0054	1	0.0	0.003
51	6.4	3.2	4.5	1.5	1	0.01	1	0.0	0.0054	1	0.0	0.003
52	6.9	3.1	4.9	1.5	1	0.01	1	0.0	0.0054	1	0.0	0.003
53	5.5	2.3	4.0	1.3	1	0.01	1	0.0	0.0054	1	0.0	0.003
54	6.5	2.8	4.6	1.5	1	0.01	1	0.0	0.0054	1	0.0	0.003
...	...	...	...	...	...	...	...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	-1	0.01	-1	0.0	0.0054	-1	0.0	0.003
146	6.3	2.5	5.0	1.9	-1	0.01	-1	0.0	0.0054	-1	0.0	0.003
147	6.5	3.0	5.2	2.0	-1	0.01	-1	0.0	0.0054	-1	0.0	0.003
148	6.2	3.4	5.4	2.3	-1	0.01	-1	0.0	0.0054	-1	0.0	0.003
149	5.9	3.0	5.1	1.8	-1	0.01	-1	0.0	0.0054	-1	0.0	0.003

100 rows × 12 columns

```
#round 3
random.seed(30)
example3 = example.sample(len(example), replace = True, weights = example['prob3'])
example3 = example3.iloc[:,0:5]
X_train = example3.iloc[0:len(iris),0:4]
y_train = example3.iloc[0:len(iris),4]

clf_gini = DecisionTreeClassifier(criterion = "gini", random_state = 100, max_depth=1)
clf = clf_gini.fit(X_train, y_train)

#adding a column pred3 after the third round of boosting
y_pred = clf_gini.predict(example.iloc[0:len(iris),0:4])
example['pred3'] = y_pred
#plotting tree for round 3 boosting
tree.plot_tree(clf)
```

```
[Text(0.5, 0.75, 'x[3] <= 1.75\ngini = 0.495\nsamples = 100\nvalue = [45.0, 55.0]'),
 Text(0.25, 0.25, 'gini = 0.408\nsamples = 77\nvalue = [22, 55]'),
 Text(0.375, 0.5, 'True  '),
 Text(0.75, 0.25, 'gini = 0.0\nsamples = 23\nvalue = [23, 0]'),
 Text(0.625, 0.5, '  False')]
```




```
#adding a field misclassified3
example.loc[example.Label != example.pred3, 'misclassified3'] = 1
example.loc[example.Label == example.pred3, 'misclassified3'] = 0
#weighted error calculation
e3 = sum(example['misclassified3'] * example['prob3']) #/len(example)
e3
```

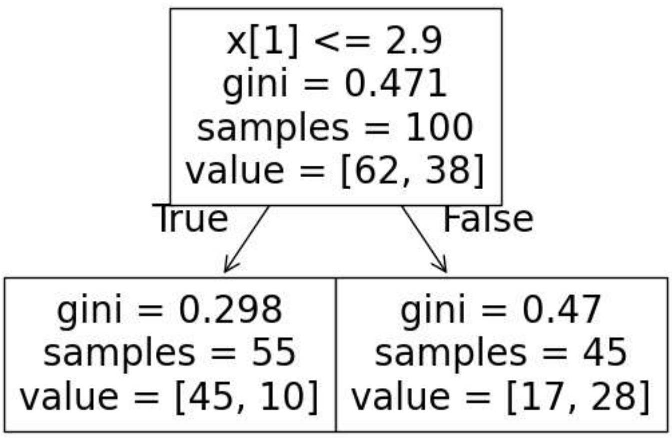
```
0.1854
```

```
#calculation of performance(alpha)
alpha3 = 0.5*log((1-e3)/e3)
#update weight
new_weight = example['prob3']*np.exp(-1*alpha3*example['Label']*example['pred3'])
z = sum(new_weight)
normalized_weight = new_weight/sum(new_weight)
example['prob4'] = round(normalized_weight,4)
example
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Label	probR1	pred1	misclassified	prob2	pred2	misclassified2	prob3	pred3	misclassified3	
50	7.0	3.2	4.7	1.4	1	0.01	1	0.0	0.0054	1	0.0	0.003	1	0.0	(
51	6.4	3.2	4.5	1.5	1	0.01	1	0.0	0.0054	1	0.0	0.003	1	0.0	(
52	6.9	3.1	4.9	1.5	1	0.01	1	0.0	0.0054	1	0.0	0.003	1	0.0	(
53	5.5	2.3	4.0	1.3	1	0.01	1	0.0	0.0054	1	0.0	0.003	1	0.0	(
54	6.5	2.8	4.6	1.5	1	0.01	1	0.0	0.0054	1	0.0	0.003	1	0.0	(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
145	6.7	3.0	5.2	2.3	-1	0.01	-1	0.0	0.0054	-1	0.0	0.003	-1	0.0	(
146	6.3	2.5	5.0	1.9	-1	0.01	-1	0.0	0.0054	-1	0.0	0.003	-1	0.0	(
147	6.5	3.0	5.2	2.0	-1	0.01	-1	0.0	0.0054	-1	0.0	0.003	-1	0.0	(
148	6.2	3.4	5.4	2.3	-1	0.01	-1	0.0	0.0054	-1	0.0	0.003	-1	0.0	(
149	5.9	3.0	5.1	1.8	-1	0.01	-1	0.0	0.0054	-1	0.0	0.003	-1	0.0	(

```
random.seed(40)
example4 = example.sample(len(example), replace = True, weights = example['prob4'])
example4 = example4.iloc[:,0:5]
X_train = example4.iloc[0:len(iris),0:4]
y_train = example4.iloc[0:len(iris),4]

clf_gini = DecisionTreeClassifier(criterion = "gini", random_state = 100, max_depth=1)
clf = clf_gini.fit(X_train, y_train)

#adding a column pred4 after the fourth round of boosting
y_pred = clf_gini.predict(example.iloc[0:len(iris),0:4])
example['pred4'] = y_pred


#plotting tree for round 4 boosting
tree.plot_tree(clf)
```

```
[Text(0.5, 0.75, 'x[1] <= 2.9\ngini = 0.471\nsamples = 100\nvalue = [62, 38]'),
 Text(0.25, 0.25, 'gini = 0.298\nsamples = 55\nvalue = [45, 10]'),
 Text(0.375, 0.5, 'True  '),
 Text(0.75, 0.25, 'gini = 0.47\nsamples = 45\nvalue = [17, 28]'),
 Text(0.625, 0.5, '  False')]
```




```
#adding a field misclassified4
example.loc[example.Label != example.pred4, 'misclassified4'] =
example.loc[example.Label == example.pred4, 'misclassified4'] =
#error calculation
```

```
e4 = sum(example['misclassified4'] * example['prob4'])
```

```
0.24179999999999988
```

```
# calculation of performance (alpha)
alpha4 = 0.5*log((1-e4)/e4)
#printing the alpha value which is used in each round of boosting
print(alpha1)
print(alpha2)
print(alpha3)
print(alpha4)
```

```
1.2211735176846021
1.1598776369434263
0.7400907710412787
0.5714181324507714
```

```
#final prediction
t = alpha1 * example['pred1'] + alpha2 * example['pred2'] + alpha3 * example['pred3'] + alpha4 * example['pred4']
#sign of the final prediction
np.sign(list(t))
```

```
array([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1., -1.,  1.,  1.,  1.,  1.,  1.,
        1., -1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1., -1., -1.,
       -1., -1., -1., -1.,  1., -1., -1., -1., -1., -1., -1., -1., -1.,
       -1., -1., -1., -1.,  1., -1., -1., -1., -1., -1., -1., -1., -1.,
       -1.,  1., -1., -1., -1.,  1.,  1., -1., -1., -1., -1., -1., -1.,
       -1., -1., -1., -1., -1., -1., -1., -1., -1.])
```

```
example['final_pred'] = np.sign(list(t))
example
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Label	probR1	pred1	misclassified	prob2	pred2	misclassified2	prob3	pred3	misclassified3
50	7.0	3.2	4.7	1.4	1	0.01	1	0.0	0.0054	1	0.0	0.003	1	0.0
51	6.4	3.2	4.5	1.5	1	0.01	1	0.0	0.0054	1	0.0	0.003	1	0.0
52	6.9	3.1	4.9	1.5	1	0.01	1	0.0	0.0054	1	0.0	0.003	1	0.0
53	5.5	2.3	4.0	1.3	1	0.01	1	0.0	0.0054	1	0.0	0.003	1	0.0
54	6.5	2.8	4.6	1.5	1	0.01	1	0.0	0.0054	1	0.0	0.003	1	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	-1	0.01	-1	0.0	0.0054	-1	0.0	0.003	-1	0.0
146	6.3	2.5	5.0	1.9	-1	0.01	-1	0.0	0.0054	-1	0.0	0.003	-1	0.0
147	6.5	3.0	5.2	2.0	-1	0.01	-1	0.0	0.0054	-1	0.0	0.003	-1	0.0
148	6.2	3.4	5.4	2.3	-1	0.01	-1	0.0	0.0054	-1	0.0	0.003	-1	0.0
149	5.9	3.0	5.1	1.8	-1	0.01	-1	0.0	0.0054	-1	0.0	0.003	-1	0.0

100 rows × 18 columns

```
#Confusion matrix
c=confusion_matrix(example['Label'], example['final_pred'])
c
```

```
array([[45,  5],
       [ 2, 48]])
```

```
#Overall Accuracy
```