

In [1]:

```
import numpy as np
import plotly.express as px
import pandas as pd
import plotly.graph_objects as go
import seaborn as sns
import matplotlib.pyplot as plt
import math
import scipy as sp
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn import datasets
from sklearn.metrics import zero_one_loss
from sklearn.model_selection import KFold
```

In [83]:

```
class SelftrainKfold:
    def __init__(self):
        self.error = []
        pass
    def readfile(self, path):
        self.raw_data = pd.read_csv(path, header=None)
        self.raw_data = self.raw_data.sample(frac=1).reset_index(drop=True)
        # shuffle the dataframe in-place and reset the index
        print(self.raw_data)
    def splitdata(self):
        self.X = self.raw_data.iloc[:, :-1] #get all columns except the last
        one
        self.y = self.raw_data.iloc[:, -1]
        self.X = self.X.to_numpy()
        self.y = self.y.to_numpy()

    def fitwithKfold(self):
        self.kfold = KFold(n_splits= 5, shuffle= True, random_state = 1)
        i=1
        for train_index, test_index in self.kfold.split(self.X):
            # print("TRAIN:", train_index, "TEST:", test_index)
```

```

X_train, X_test = self.X[train_index], self.X[test_index]
y_train, y_test = self.y[train_index], self.y[test_index]

## 15% of the training data is labeled
X_train_lab, X_unl, y_train_lab, _ = train_test_split(X_train,
y_train, train_size=0.15, random_state=1)
# 1. Training on the labeled dataset
print("Fold_{}".format(i))
while True:
    #Step1
    clf = svm.SVC(kernel='linear',
probability=True,C=1).fit(X_train_lab, y_train_lab)
    print ('Accuracy: ',clf.score(X_test, y_test), ' error: ',
1 - clf.score(X_test, y_test) )
    if (len(X_unl) == 0): # exit the training if no more
unlabeled data
        #                print("fist break")
        break
    #Step2
    clp= clf.predict_proba(X_unl)
    clf_prob = pd.DataFrame(clp)
    lab=clf.predict(X_unl)
    clf_prob["max"] = clf_prob.max(axis = 1)
    clf_prob["label"] = lab
    #    print (clf_prob)
    th = 0.8
    if len(X_unl[clf_prob["max"] > th]) == 0: #exit if no more
samples meets the threshold condition
        break
    #Step3
    #add the predicted labels to the training dataset
    X_train_new = np.append(X_train_lab, X_unl[clf_prob["max"]
> th], axis=0)
    y_train_new = np.append(y_train_lab, clf_prob['label']
[clf_prob["max"] > th].values, axis=0)
    X_train_lab = X_train_new

```

```

y_train_lab = y_train_new

#remove the added labels from the unlabeled dataset
X_unl_df = pd.DataFrame(X_unl)
X_unl_df = X_unl_df.drop(X_unl_df[clf_prob["max"] >
th].index).reset_index(drop=True)

#update the unlabeled set
X_unl = X_unl_df.values

print("Fold_{}_done".format(i))
self.error.append(clf.score(X_test, y_test))
i+=1

def ave_error(self):
    return sum(self.error)/len(self.error)

```

In [84]:

```
obj = SelftrainKfold()
```

In [85]:

```
obj.readfile("/home/safwan/Documents/spring2021/ece523/hw/hw5/breast-
cancer.csv")
```

	0	1	2	3	4	5	6	\
0	-0.656576	0.912871	0.531300	-0.455814	0.508429	-1.42093	-0.937281	
1	0.331744	-0.912871	2.430700	-0.455814	0.508429	-1.42093	1.063180	
2	-1.644900	0.912871	1.481000	0.413271	0.508429	1.28831	1.063180	
3	0.331744	-0.912871	0.531300	0.413271	0.508429	1.28831	-0.937281	
4	0.331744	-0.912871	-0.893249	-0.455814	0.508429	-0.06631	1.063180	
...
281	0.331744	-0.912871	-0.418399	-0.455814	0.508429	-1.42093	1.063180	
282	-0.656576	0.912871	1.006150	-0.455814	-1.511170	1.28831	1.063180	
283	1.320060	-0.912871	-1.368100	-0.455814	0.508429	-0.06631	1.063180	
284	-0.656576	0.912871	0.056451	-0.455814	0.508429	-0.06631	-0.937281	
285	-0.656576	0.912871	0.056451	-0.455814	0.508429	-0.06631	1.063180	
...
	7	8	9					
0	-0.130877	0.557527	0					
1	0.700918	0.557527	0					
2	0.700918	-1.787360	0					
3	-0.130877	0.557527	1					
4	0.700918	0.557527	0					
...
281	-0.130877	0.557527	0					
282	-0.130877	-1.787360	0					
283	-0.962672	-1.787360	0					
284	-0.962672	-1.787360	0					
285	-0.130877	0.557527	1					

[286 rows x 10 columns]

In [86]: `obj.splitdata()`

In [87]: `obj.fitwithKfold()`

```
Fold_1
Accuracy: 0.7758620689655172 error: 0.22413793103448276
Accuracy: 0.7758620689655172 error: 0.22413793103448276
Accuracy: 0.7758620689655172 error: 0.22413793103448276
Accuracy: 0.7758620689655172 error: 0.22413793103448276
Accuracy: 0.7413793103448276 error: 0.2586206896551724
Accuracy: 0.7758620689655172 error: 0.22413793103448276
Accuracy: 0.7758620689655172 error: 0.22413793103448276
Accuracy: 0.7758620689655172 error: 0.22413793103448276
Accuracy: 0.7586206896551724 error: 0.24137931034482762
Accuracy: 0.7586206896551724 error: 0.24137931034482762
Fold_1_done
Fold_2
Accuracy: 0.5087719298245614 error: 0.49122807017543857
Accuracy: 0.5087719298245614 error: 0.49122807017543857
Accuracy: 0.5087719298245614 error: 0.49122807017543857
Accuracy: 0.5263157894736842 error: 0.4736842105263158
Accuracy: 0.5263157894736842 error: 0.4736842105263158
Accuracy: 0.5263157894736842 error: 0.4736842105263158
Fold_2_done
Fold_3
Accuracy: 0.8070175438596491 error: 0.19298245614035092
Accuracy: 0.7894736842105263 error: 0.21052631578947367
Accuracy: 0.7894736842105263 error: 0.21052631578947367
Fold_3_done
Fold_4
Accuracy: 0.6666666666666666 error: 0.33333333333333337
Accuracy: 0.6666666666666666 error: 0.33333333333333337
Accuracy: 0.6842105263157895 error: 0.3157894736842105
Accuracy: 0.6842105263157895 error: 0.3157894736842105
Accuracy: 0.7192982456140351 error: 0.2807017543859649
Accuracy: 0.7017543859649122 error: 0.29824561403508776
Accuracy: 0.6491228070175439 error: 0.3508771929824561
Accuracy: 0.7368421052631579 error: 0.26315789473684215
Accuracy: 0.7368421052631579 error: 0.26315789473684215
Fold_4_done
Fold_5
Accuracy: 0.47368421052631576 error: 0.5263157894736843
Accuracy: 0.47368421052631576 error: 0.5263157894736843
Accuracy: 0.47368421052631576 error: 0.5263157894736843
Accuracy: 0.47368421052631576 error: 0.5263157894736843
Accuracy: 0.47368421052631576 error: 0.5263157894736843
Accuracy: 0.543859649122807 error: 0.45614035087719296
Accuracy: 0.631578947368421 error: 0.368421052631579
Accuracy: 0.6491228070175439 error: 0.3508771929824561
Accuracy: 0.631578947368421 error: 0.368421052631579
Accuracy: 0.631578947368421 error: 0.368421052631579
Accuracy: 0.6491228070175439 error: 0.3508771929824561
Accuracy: 0.631578947368421 error: 0.368421052631579
Fold_5_done
```

In []:

In []:

In [88]: `obj.ave_error()`

Out[88]: 0.6885662431941924

For the results above, we can see that the self training did help in 3 out of 5 folds and the boost in performance was not significant. However, we can also observe that in some cases the self training algorithm gave worst accuracy compared to the previous iteration.

In [90]: `# Using different dataset`
`obj1 = SelftrainKfold()`
`obj1.readfile('/home/safwan/Documents/spring2021/ece523/hw/hw5/abalone.csv')`

	0	1	2	3	4	5	6	\
0	0.053792	0.882716	0.928243	0.489721	0.797852	0.590786	1.089330	
1	1.261790	-1.781890	-1.842820	-1.542460	-1.373890	-1.326200	-1.309990	
2	1.261790	-1.823520	-1.893200	-1.662000	-1.402440	-1.339720	-1.364730	
3	0.053792	0.466371	0.525180	0.250642	0.584756	0.892638	0.409677	
4	-1.154210	1.049250	1.079390	1.087420	1.112910	1.759900	-0.000854	
...	
4172	-1.154210	1.299060	1.331310	1.206960	1.619650	1.608970	1.581970	
4173	-1.154210	1.757040	1.784750	1.087420	2.641290	2.669960	3.370060	
4174	0.053792	0.882716	0.877860	1.446040	1.216910	1.349920	1.417760	
4175	1.261790	-0.324683	-0.230565	-0.227518	-0.531705	-0.490476	-0.023661	
4176	-1.154210	1.299060	1.331310	1.326500	1.944900	0.888133	1.303720	
...	
	7	8						
0	0.762695	2						
1	-1.453500	0						
2	-1.449910	0						
3	0.367587	0						
4	0.935105	2						
...	
4172	1.671440	2						
4173	1.876180	2						
4174	1.014130	2						
4175	-0.609405	0						
4176	1.362540	2						

[4177 rows x 9 columns]

In [91]: `obj1.splitdata()`
`obj1.fitwithKfold()`

Fold_1

Accuracy:	0.645933014354067	error:	0.354066985645933
Accuracy:	0.645933014354067	error:	0.354066985645933
Accuracy:	0.645933014354067	error:	0.354066985645933
Accuracy:	0.6423444976076556	error:	0.35765550239234445
Accuracy:	0.6495215311004785	error:	0.3504784688995215
Accuracy:	0.6435406698564593	error:	0.3564593301435407
Accuracy:	0.6399521531100478	error:	0.36004784688995217
Accuracy:	0.638755980861244	error:	0.36124401913875603
Accuracy:	0.6399521531100478	error:	0.36004784688995217
Accuracy:	0.6351674641148325	error:	0.3648325358851675

Accuracy:	0.6339712918660287	error:	0.36602870813397126
Accuracy:	0.6339712918660287	error:	0.36602870813397126
Accuracy:	0.6351674641148325	error:	0.3648325358851675
Fold_1_done			
Fold_2			
Accuracy:	0.6435406698564593	error:	0.3564593301435407
Accuracy:	0.6435406698564593	error:	0.3564593301435407
Accuracy:	0.6447368421052632	error:	0.35526315789473684
Accuracy:	0.6507177033492823	error:	0.34928229665071775
Accuracy:	0.6363636363636364	error:	0.36363636363636365
Accuracy:	0.6196172248803827	error:	0.38038277511961727
Accuracy:	0.6196172248803827	error:	0.38038277511961727
Accuracy:	0.6172248803827751	error:	0.3827751196172249
Accuracy:	0.618421052631579	error:	0.381578947368421
Accuracy:	0.6172248803827751	error:	0.3827751196172249
Accuracy:	0.6136363636363636	error:	0.38636363636363635
Accuracy:	0.6100478468899522	error:	0.38995215311004783
Accuracy:	0.6100478468899522	error:	0.38995215311004783
Accuracy:	0.6088516746411483	error:	0.3911483253588517
Accuracy:	0.611244019138756	error:	0.38875598086124397
Accuracy:	0.6064593301435407	error:	0.3935406698564593
Accuracy:	0.6052631578947368	error:	0.39473684210526316
Accuracy:	0.6028708133971292	error:	0.3971291866028708
Accuracy:	0.6016746411483254	error:	0.39832535885167464
Accuracy:	0.6016746411483254	error:	0.39832535885167464
Accuracy:	0.5992822966507177	error:	0.40071770334928225
Accuracy:	0.5992822966507177	error:	0.40071770334928225
Accuracy:	0.5980861244019139	error:	0.4019138755980861
Fold_2_done			
Fold_3			
Accuracy:	0.6347305389221557	error:	0.3652694610778443
Accuracy:	0.6347305389221557	error:	0.3652694610778443
Accuracy:	0.6215568862275449	error:	0.3784431137724551
Accuracy:	0.6239520958083832	error:	0.3760479041916168
Accuracy:	0.6107784431137725	error:	0.3892215568862275
Accuracy:	0.6023952095808384	error:	0.39760479041916164
Accuracy:	0.6071856287425149	error:	0.3928143712574851
Accuracy:	0.6059880239520958	error:	0.39401197604790417
Accuracy:	0.6059880239520958	error:	0.39401197604790417
Accuracy:	0.6071856287425149	error:	0.3928143712574851
Accuracy:	0.6035928143712574	error:	0.39640718562874255
Accuracy:	0.6047904191616766	error:	0.39520958083832336
Accuracy:	0.6047904191616766	error:	0.39520958083832336
Fold_3_done			
Fold_4			
Accuracy:	0.629940119760479	error:	0.37005988023952097
Accuracy:	0.629940119760479	error:	0.37005988023952097
Accuracy:	0.6347305389221557	error:	0.3652694610778443
Accuracy:	0.6275449101796408	error:	0.37245508982035924
Accuracy:	0.6179640718562874	error:	0.38203592814371257
Accuracy:	0.6095808383233533	error:	0.3904191616766467
Accuracy:	0.6	error:	0.4
Accuracy:	0.5964071856287425	error:	0.4035928143712575
Accuracy:	0.5976047904191617	error:	0.4023952095808383
Accuracy:	0.5964071856287425	error:	0.4035928143712575
Accuracy:	0.5976047904191617	error:	0.4023952095808383
Accuracy:	0.5952095808383233	error:	0.4047904191616767
Accuracy:	0.5964071856287425	error:	0.4035928143712575
Accuracy:	0.5964071856287425	error:	0.4035928143712575
Accuracy:	0.5976047904191617	error:	0.4023952095808383
Accuracy:	0.5976047904191617	error:	0.4023952095808383
Fold_4_done			
Fold_5			
Accuracy:	0.6491017964071857	error:	0.3508982035928143
Accuracy:	0.6491017964071857	error:	0.3508982035928143

```
Accuracy: 0.644311377245509 error: 0.355688622754491
Accuracy: 0.6431137724550898 error: 0.3568862275449102
Accuracy: 0.6479041916167665 error: 0.3520958083832335
Accuracy: 0.6479041916167665 error: 0.3520958083832335
Accuracy: 0.6419161676646706 error: 0.35808383233532937
Accuracy: 0.6383233532934132 error: 0.36167664670658684
Accuracy: 0.6407185628742516 error: 0.35928143712574845
Accuracy: 0.6383233532934132 error: 0.36167664670658684
Accuracy: 0.6395209580838324 error: 0.36047904191616764
Accuracy: 0.6407185628742516 error: 0.35928143712574845
Accuracy: 0.6407185628742516 error: 0.35928143712574845
Fold_5_done
```

In [93]:

```
obj1.ave_error()
```

Out[93]: 0.6152734721943673

We see similar observation to the previous results

In []: