

**NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY**

(AN AUTONOMOUS INSTITUTION AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM)

**Machine learning**

**7th semester Sec A**

LA-2 Assignment

Submitted to:

Mr. Aditya Shastry

Submitted by:

D. Jagath Singh

**USN:1NT17IS042**

Darshit Sinha

**USN:1NT17IS044**

**Question:**

Parkinson Disease Prediction Using Machine Learning: Parkinson’s disease is one of the main neurological disorders affecting the aged. It is an environmentally influenced, neurodegenerative disease that is characterized by tremors, stiffness of limbs and trunk, slowness of voluntary movements called Bradykinesia, postural instability and a distinctive shuffling gait with a stooped position called “Parkinson’s gait”. According to one estimate more than 1% of the population suffers from Parkinson’s disease after reaching 55 years of age. Based on these points, the main aim of the project is to develop a ML algorithm that can predict the Parkinson’s disease. The input parameters that impact the Parkinson’s disease are utilized to forecast whether the person has Parkinson or not.

**Dataset info:**

**Obtained from:**

Erdogdu Sakar, B., Isenkul, M., Sakar, C.O., Sertbas, A., Gurgen, F., Delil, S., Apaydin, H., Kursun, O., 'Collection and Analysis of a Parkinson Speech Dataset with Multiple Types of Sound Recordings', IEEE Journal of Biomedical and Health Informatics, vol. 17(4), pp. 828-834, 2013.

2 files – train, test

**Train:**

Audio recordings of 20 patients(14 M, 6 F) having PD and 20 without PD

Each subject – 26 audio files

1: sustained vowel (aaaâ€¦â€¦)  
2: sustained vowel (oooâ€¦...)  
3: sustained vowel (uuuâ€¦...)  
4-13: numbers from 1 to 10  
14-17: short sentences  
18-26: words

Total : 40 patients \* 26 audio files \* 29 features

**Feature information:**

column 1: Subject id  
colum 2-27: features  
features 1-5: Jitter (local),Jitter (local, absolute),Jitter (rap),Jitter (ppq5),Jitter (ddp),  
features 6-11: Shimmer (local),Shimmer (local, dB),Shimmer (apq3),Shimmer (apq5), Shimmer (apq11),Shimmer (dda),  
features 12-14: AC,NTH,HTN,  
features 15-19: Median pitch,Mean pitch,Standard deviation,Minimum pitch,Maximum pitch,  
features 20-23: Number of pulses,Number of periods,Mean period,Standard deviation of period, features 24-26: Fraction of locally unvoiced frames,Number of voice breaks, Degree of voice breaks  
column 28: UPDRS  
column 29: class information

**Test:**

28 subjects, all having Parkinson disease, since we are more concerned about true positives.

6 audio clips of each subject:

1-3: sustained vowel (aaaâ€¦â€¦)  
4-6: sustained vowel (oooâ€¦â€¦)

Features:

Same as test data, but no updrs.

**Methodology:**

1.Preprocessing:

ID not useful for learning. Remove.

Updrs cannot be obtained using only app. Remove since it doesn’t give useful info for testing, and updrs is also missing in the provided test data, so no point giving null values for updrs. We can just remove it from training.

2**. Algorithm:**

NuSVC:

In scikit SVC and nuSVC are mathematically equivalent with both methods based on the library libsvm. The main difference is that SVC uses the parameter C while nuSVC uses the parameter nu.

The problem with the parameter C is:

1. That it can take any positive value
2. That it has no direct interpretation.

It is therefore hard to choose correctly and one must resort to cross validation or direct experimentation to find a suitable value.

In response Schölkopf et al. reformulated SVM to take a new regularization parameter nu. This parameter is:

1. Bounded between 0 and 1.
2. Has a direct interpretation.

The parameter nu is an upper bound on the fraction of margin errors and a lower bound of the fraction of support vectors relative to the total number of training examples. For example, if you set it to 0.05 you are guaranteed to find at most 5% of your training examples being misclassified (at the cost of a small margin, though) and at least 5% of your training examples being support vectors.

**Non liner support vector machine:**

In machine learning, support-vector machines (SVMs, also support-vector networks) are supervised learning models with associated learning algorithms that analyse data for classification and regression analysis.

The original maximum-margin hyperplane algorithm proposed by Vapnik in 1963 constructed a linear classifier. However, in 1992, Bernhard Boser, Isabelle Guyon and Vladimir Vapnik suggested a way to create nonlinear classifiers by applying the kernel trick (originally proposed by Aizerman et al.) to maximum-margin hyperplanes. The resulting algorithm is formally similar, except that every dot product is replaced by a nonlinear kernel function. This allows the algorithm to fit the maximum-margin hyperplane in a transformed feature space. The transformation may be nonlinear and the transformed space high-dimensional; although the classifier is a hyperplane in the transformed feature space, it may be nonlinear in the original input space.

Models:

**Classifier clf** – trained on the original dataset, with each subject having 26 different audio files i.e, trained on 26\*40 = 1040 records - NuSVC, with rbf kernel and gamma automatically assigned..

Test accuracy is found to be really low, 66%..

Even after using majority voting rather than individual audio clips, test accuracy is found to be 75%.

**Classifier clf1** – this classifier is trained on summarized dataset..

All the data model is trained on is now classified correctly and the model only makes one mistake in one of the audio clip of test data. The model performs a lot better combining the audio clips to one. - NuSVC, with rbf kernel and gamma automatically assigned.

**Summarising data:**

For each feature in dataset:

Divide feature such that each part has all the audio recordings of a single subject.

Find ['mean','median','trim10','trim25','std','iqr','mad'] for each of these parts.

So, with this, we basically combine all audio recordings of a subject to one record. So, each subject will have a single record. At the same time, we are calculating ['mean','median','trim10','trim25','std','iqr','mad'] – 7 parameters for each feature. So the no. of features will increase.

So, if the dataset was 1040\*26, the new summarized dataset would be 40\*(26\*7)

**Source code:**

"""

Parkinson disease classification

Dataset by: Erdogdu Sakar, B., Isenkul, M., Sakar, C.O., Sertbas, A., Gurgen, F., Delil, S., Apaydin, H., Kursun, O.,

'Collection and Analysis of a Parkinson Speech Dataset with Multiple Types of Sound Recordings',

IEEE Journal of Biomedical and Health Informatics, vol. 17(4), pp. 828-834, 2013.

Author: Jagath, Darshit

"""

from sklearn.model\_selection import LeaveOneOut

import pandas as pd

import numpy as np

from sklearn.svm import SVC,NuSVC

from sklearn.preprocessing import StandardScaler

from sklearn.pipeline import make\_pipeline

import scipy

import pickle as pkl

from sklearn.metrics import confusion\_matrix

#to seperate the data of each subject

def split\_data(data,n,split\_size,column = None):

    if column != None:

        d = np.array(data[column])

    else:

        d = np.array(data)

    data\_split = []

    group = []

    index = 0

    while(index < n):

        group.append(d[index])

        if index % split\_size == split\_size - 1:

            data\_split.append(group)

            group = []

        index += 1

    return data\_split

def summarize\_data(data, rowc, columns, split\_size):

    calculate = ['mean','median','trim10','trim25','std','iqr','mad']#mad - mean absolute deviation, iqr - interquartile range

    df = pd.DataFrame()

    for i in columns:

        data\_split = split\_data(data,rowc,split\_size,i)

        df['mean\_'+i] = np.mean(data\_split,axis = 1)

        df['median\_'+i] = np.median(data\_split,axis = 1)

        df['trim10\_'+i] = scipy.stats.trim\_mean(data\_split, 0.1, axis = 1)

        df['trim25\_'+i] = scipy.stats.trim\_mean(data\_split, 0.25, axis = 1)

        df['std\_'+i] = np.std(data\_split,axis = 1)

        df['iqr\_'+i] = scipy.stats.iqr(data\_split, axis = 1)

        df['mad\_'+i] = scipy.stats.median\_abs\_deviation(data\_split,axis = 1)

    return df

#read train data

df = pd.read\_csv(r"C:\Users\jagat\Downloads\train\_data.txt")

column\_list = list(df.iloc[:,1:-2].columns)

X\_train,Y\_train = np.array(df.iloc[:,1:-2]),np.array(df.iloc[:,-1])

train\_data\_len = len(X\_train)

print('feature list:')

print(column\_list)

print('\n')

print('sample train data(one subject)(one subject and 4 features only:')

print(df.iloc[:26,:5])

print()

#read test data

df1 = pd.read\_csv(r"C:\Users\jagat\Downloads\test\_data.txt")

X\_test,Y\_test = np.array(df1.iloc[:,1:-1]),np.array(df1.iloc[:,-1])

test\_data\_len = len(X\_test)

print('sample test data:(one subject and 4 features only)')

print(df1.iloc[:6,:5])

print()

#create and train clasifier

clf = make\_pipeline(StandardScaler(), NuSVC(gamma='auto'))#standard scaler makes mean = 0 and variance = 1

#clf = NuSVC(gamma='auto') #causes overfitting, if done without scaling

clf.fit(X\_train,Y\_train)

#predict

test\_pred\_Y = clf.predict(X\_test)

#accuracies of individual audio

print("individually test accuracy without summarizing(individual samples):"+str(clf.score(X\_test,Y\_test)))

print("individually train accuracy without summarizing(individual samples):"+str(clf.score(X\_train,Y\_train)))

#accuracy subject-wise

grouped\_test\_pred\_Y = split\_data(test\_pred\_Y,test\_data\_len,6,None)

test\_pred\_Y\_ForEachSubject = []

for i in grouped\_test\_pred\_Y:

    count = 0

    for j in i:

        if j == 1:

            count += 1

    if count >= 3:

        test\_pred\_Y\_ForEachSubject.append(1)

    else:

        test\_pred\_Y\_ForEachSubject.append(0)

print('\n')

print('accuracy using non linear svm without summarizing is:(considering all classified audio clips of a subject, by maximum voting)')

count = 0

for i in test\_pred\_Y\_ForEachSubject:

    if i == 1:

        count += 1

print(count/len(test\_pred\_Y\_ForEachSubject))

print('\n')

#train with summarizing

df\_summ =  summarize\_data(df, train\_data\_len, column\_list, 26)

print('data after summarizing:(only 3 features shown)')

print(df\_summ.iloc[:,:3])

#add class label

label = [1]\*20

label.extend([0]\*20)

df\_summ['class'] = label

df\_summ.to\_csv(r'C:\Users\jagat\Downloads\summ\_train\_data.csv',index=False)

X\_train,Y\_train = np.array(df\_summ.iloc[:,:-1]),np.array(df\_summ.iloc[:,-1])

clf1 = make\_pipeline(StandardScaler(), NuSVC(gamma='auto'))

clf1.fit(X\_train,Y\_train)

print('\n\n')

print('train set accuracy after summarizing:(By using maximum voting) '+str(clf1.score(X\_train,Y\_train)))

#read and summarize test data:

df\_summ\_test = summarize\_data(df1,test\_data\_len,column\_list,6)

df\_summ\_test['class'] = [1]\*28

df\_summ\_test.to\_csv(r'C:\Users\jagat\Downloads\summ\_test\_data.csv',index=False)

X\_test,Y\_test = np.array(df\_summ\_test.iloc[:,:-1]),np.array(df\_summ\_test.iloc[:,-1])

print('test set accuracy after summarizing: '+str(clf1.score(X\_test,Y\_test)))

print('actual classes of test data: ')

print(Y\_test)

print('predicted classes of test data: ')

print(clf1.predict(X\_test))

#save model in pickle file

pkl.dump(clf1, open(r'C:\Users\jagat\Downloads\summ\_test\_data.csv', 'wb'))

Output:

feature list:

['Jitter (local)', 'Jitter (local, absolute)', 'Jitter (rap)', 'Jitter (ppq5)', 'Jitter (ddp)', 'Shimmer (local)', 'Shimmer (local, dB)', 'Shimmer (apq3)', 'Shimmer (apq5)', ' Shimmer (apq11)', 'Shimmer (dda)', 'AC', 'NTH', 'HTN', ' Median pitch', 'Mean pitch', 'Standard deviation', 'Minimum pitch', 'Maximum pitch', 'Number of pulses', 'Number of periods', 'Mean period', 'Standard deviation of period', 'Fraction of locally unvoiced frames', 'Number of voice breaks', 'Degree of voice breaks']

sample train data(one subject)(one subject and 4 features only:

id Jitter(local) Jitter(local,absolute) Jitter (rap) Jitter (ppq5)

0 1 1.488 0.000090 0.900 0.794

1 1 0.728 0.000038 0.353 0.376

2 1 1.220 0.000074 0.732 0.670

3 1 2.502 0.000123 1.156 1.634

4 1 3.509 0.000167 1.715 1.539

5 1 2.470 0.000126 1.358 1.341

6 1 1.583 0.000082 0.768 0.864

7 1 1.920 0.000099 0.926 1.031

8 1 2.257 0.000119 1.239 1.530

9 1 1.594 0.000085 0.850 0.951

10 1 2.286 0.000122 1.081 1.257

11 1 1.728 0.000093 0.807 1.056

12 1 2.105 0.000110 0.902 1.076

13 1 3.343 0.000164 1.349 1.826

14 1 2.025 0.000099 1.028 1.251

15 1 2.517 0.000118 1.160 1.350

16 1 3.290 0.000149 1.567 1.822

17 1 5.534 0.000258 2.380 3.171

18 1 1.137 0.000054 0.473 0.608

19 1 2.162 0.000099 1.217 1.059

20 1 1.533 0.000076 0.462 0.340

21 1 2.783 0.000135 1.221 1.338

22 1 3.999 0.000188 1.928 2.231

23 1 2.227 0.000106 1.014 1.418

24 1 1.728 0.000079 0.811 0.820

25 1 2.638 0.000124 1.418 1.477

sample test data:(one subject and 4 features only)

id Jitter(local) Jitter(local,absolute) Jitter(rap) Jitter (ppq5)

0 1 0.135 0.000007 0.067 0.078

1 1 0.143 0.000007 0.073 0.081

2 1 0.162 0.000008 0.087 0.089

3 1 0.140 0.000007 0.075 0.089

4 1 0.150 0.000007 0.080 0.097

5 1 0.208 0.000009 0.077 0.095

individually test accuracy without summarizing(individual samples):0.6607142857142857

individually train accuracy without summarizing(individual samples):0.8740384615384615

accuracy using non linear svm without summarizing is:(considering all classified audio clips of a subject, by maximum voting)

0.75

data after summarizing:(only 3 features shown)

mean\_Jitter (local) median\_Jitter (local) trim10\_Jitter (local)

0 2.319462 2.1945 2.223091

1 2.688038 2.7965 2.710364

2 3.006423 2.6255 2.722545

3 1.545038 1.3485 1.500182

4 2.749600 2.7688 2.665391

5 2.797154 2.8180 2.843591

6 2.760500 2.6940 2.692682

7 2.216308 2.3140 2.191455

8 2.032962 1.8750 2.008318

9 3.215538 2.8265 3.029636

10 2.390885 2.1180 2.324682

11 4.713923 4.8765 4.632409

12 2.170654 2.2685 2.158818

13 3.159962 3.4515 3.251227

14 2.772615 3.0430 2.750955

15 2.557615 2.7925 2.584091

16 2.054577 2.0475 2.055727

17 5.335346 4.9900 5.224091

18 3.667577 3.8745 3.664273

19 2.878308 2.3515 2.757409

20 2.222692 2.2530 2.212364

21 2.185038 2.0965 2.106955

22 3.067423 2.9785 3.004545

23 1.625462 1.4725 1.627727

24 2.453462 2.5155 2.409227

25 3.263385 2.8770 3.097091

26 2.810692 2.7435 2.578455

27 1.753692 1.7680 1.745091

28 2.755346 2.6140 2.712182

29 2.019538 1.9705 1.975955

30 3.398269 2.2225 3.140182

31 4.230962 2.2595 3.666545

32 3.036077 1.8755 2.455273

33 2.234308 2.2515 2.180909

34 2.104654 1.8150 2.050136

35 1.413885 1.2960 1.381182

36 2.108385 2.3430 2.098455

37 2.386731 2.5120 2.435682

38 2.098962 2.2165 1.977227

39 2.979462 2.7720 2.881318

train set accuracy after summarizing:(By using maximum voting) 1.0

test set accuracy after summarizing: 0.9642857142857143

actual classes of test data:

[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]

predicted classes of test data:

[1 1 0 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]