In [1]:

```
# Task 1.1: Data Retrieving
# Importing pandas, numpy and pyplot packages
import pandas as pd
import numpy as np
import random
import matplotlib.pyplot as plt

# Importing StarWars.csv dataset and assigning it to a variable using read_csv function
mice_file = 'Data_Cortex_Nuclear.csv'
mice = pd.read_csv(mice_file, sep=',', decimal = '.')

# the top row contains the response options from the survey
mice.head(3)
```

Out[1]:

	MouseID	DYRK1A_N	ITSN1_N	BDNF_N	NR1_N	NR2A_N	pAKT_N	pBRAF_N	pCAMKII
0	309_1	0.503644	0.747193	0.430175	2.816329	5.990152	0.218830	0.177565	2.3737
1	309_2	0.514617	0.689064	0.411770	2.789514	5.685038	0.211636	0.172817	2.2921
2	309_3	0.509183	0.730247	0.418309	2.687201	5.622059	0.209011	0.175722	2.2833

3 rows × 82 columns

In [2]:

```
mice.shape
```

Out[2]:

```
(1080, 82)
```

In [3]:

```
mice.isna().sum().sum()
```

Out[3]:

```
1396
```

In [4]:

```
mice.dtypes
```

Out[4]:

```
MouseID       object
DYRK1A_N     float64
ITSN1_N      float64
BDNF_N       float64
NR1_N        float64
              ...
CaNA_N       float64
Genotype      object
Treatment     object
Behavior      object
class         object
Length: 82, dtype: object
```

In [5]:

```
# For-loop for checking the typos by using unqiue value counts in each column
for i in mice.columns:
    m = mice[i].value_counts(dropna = False)
    print(m)
    print()
```

```
3534_2      1
362_1       1
3414_4      1
3412_15     1
3507_15     1
           ..
3476_3      1
3426_1      1
3423_8      1
320_6       1
50810D_8    1
Name: MouseID, Length: 1080, dtype: int64

NaN         3
0.399414    1
0.388473    1
0.599745    1
0.356944    1
           ..
0.429252    1
0.194417    1
0.481273    1
0.229960    1
0.343750    1
Name: DYRK1A_N, Length: 1078, dtype: int64

NaN         3
0.639280    2
0.481460    1
1.025424    1
0.349683    1
           ..
0.572398    1
0.540086    1
0.477591    1
1.051810    1
0.460938    1
Name: ITSN1_N, Length: 1077, dtype: int64

NaN         3
0.302132    1
0.315109    1
0.341103    1
0.331943    1
           ..
0.237249    1
0.288585    1
0.394664    1
0.279554    1
0.294870    1
Name: BDNF_N, Length: 1078, dtype: int64

NaN         3
```

```
2.164182    1
1.912610    1
2.704669    1
2.384171    1
           ..
1.936624    1
2.930717    1
2.051009    1
2.425020    1
1.875000    1
Name: NR1_N, Length: 1078, dtype: int64

NaN         3
4.036498    1
3.560770    1
4.507068    1
4.102347    1
           ..
2.582260    1
6.215859    1
5.256264    1
2.978824    1
4.500000    1
Name: NR2A_N, Length: 1078, dtype: int64

NaN         3
0.231177    2
0.234393    1
0.246624    1
0.200131    1
           ..
0.253184    1
0.196059    1
0.237969    1
0.219771    1
0.168654    1
Name: pAKT_N, Length: 1077, dtype: int64

NaN         3
0.204209    2
0.210526    2
0.199140    1
0.178889    1
           ..
0.134006    1
0.201903    1
0.171334    1
0.173021    1
0.200657    1
Name: pBRAF_N, Length: 1076, dtype: int64

NaN         3
2.586144    1
2.512737    1
4.503084    1
2.362741    1
           ..
4.652908    1
3.760677    1
2.248283    1
6.288912    1
```

```
2.154635    1
Name: pCAMKII_N, Length: 1078, dtype: int64

NaN         3
0.147956    1
0.213052    1
0.150437    1
0.263736    1
           ..
0.146601    1
0.200000    1
0.175000    1
0.218104    1
0.206526    1
Name: pCREB_N, Length: 1078, dtype: int64

NaN         3
1.550766    1
1.800946    1
1.171334    1
2.004403    1
           ..
1.151282    1
1.192464    1
1.505378    1
1.358888    1
1.000000    1
Name: pELK_N, Length: 1078, dtype: int64

NaN         3
0.298799    1
0.396954    1
0.414409    1
0.685744    1
           ..
0.936017    1
0.384520    1
0.554367    1
0.349242    1
0.648438    1
Name: pERK_N, Length: 1078, dtype: int64

NaN         3
0.333333    2
0.320303    1
0.264583    1
0.333716    1
           ..
0.374258    1
0.333051    1
0.246769    1
0.252390    1
0.299086    1
Name: pJNK_N, Length: 1077, dtype: int64

NaN         3
0.328137    1
0.312396    1
0.427391    1
0.368855    1
           ..
```

```
0.268399    1
0.287196    1
0.339076    1
0.249818    1
0.318359    1
Name: PKCA_N, Length: 1078, dtype: int64

NaN         3
0.289109    1
0.248998    1
0.300158    1
0.276591    1
           ..
0.181978    1
0.281399    1
0.261017    1
0.233021    1
0.308158    1
Name: pMEK_N, Length: 1078, dtype: int64

NaN         3
1.060232    1
0.869152    1
0.987288    1
0.827275    1
           ..
0.735497    1
0.850015    1
0.751472    1
0.740304    1
0.756207    1
Name: pNR1_N, Length: 1078, dtype: int64

NaN         3
0.834059    1
0.771915    1
0.661363    1
0.735433    1
           ..
0.640845    1
0.423113    1
0.935287    1
0.792998    1
0.781250    1
Name: pNR2A_N, Length: 1078, dtype: int64

NaN         3
1.548731    1
1.915403    1
1.495832    1
1.945114    1
           ..
1.355399    1
1.682106    1
1.261149    1
1.344581    1
1.389278    1
Name: pNR2B_N, Length: 1078, dtype: int64

NaN         3
1.193504    1
```

```
1.400877    1
1.968275    1
1.040727    1
           ..
1.301867    1
1.252248    1
1.240528    1
2.242223    1
1.110085    1
Name: pPKCAB_N, Length: 1078, dtype: int64

NaN         3
0.566439    1
0.461612    1
0.365742    1
0.514658    1
           ..
0.502687    1
0.422512    1
0.510545    1
0.392021    1
0.500000    1
Name: pRSK_N, Length: 1078, dtype: int64

NaN         3
0.478502    1
0.609620    1
0.530597    1
0.759702    1
           ..
0.606662    1
0.732022    1
0.515954    1
0.626361    1
0.625000    1
Name: AKT_N, Length: 1078, dtype: int64

NaN         3
0.238042    1
0.316123    1
0.366388    1
0.252137    1
           ..
0.385990    1
0.282400    1
0.749467    1
0.244735    1
1.662267    1
Name: BRAF_N, Length: 1078, dtype: int64

NaN         3
0.389649    1
0.394527    1
0.323570    1
0.212960    1
           ..
0.369572    1
0.354898    1
0.404830    1
0.324267    1
0.437500    1
```

```
Name: CAMKII_N, Length: 1078, dtype: int64

NaN         3
0.210938    2
0.228852    2
0.186047    2
0.197880    2
           ..
0.176790    1
0.191846    1
0.155029    1
0.186129    1
0.185197    1
Name: CREB_N, Length: 1074, dtype: int64

NaN         18
1.360116     1
1.840786     1
1.801658     1
1.401744     1
            ..
0.989746     1
0.926516     1
0.999687     1
0.907672     1
1.064129     1
Name: ELK_N, Length: 1063, dtype: int64

NaN         3
1.507803    1
2.386324    1
2.169929    1
3.048822    1
           ..
2.209994    1
1.930710    1
3.098614    1
2.220048    1
2.500000    1
Name: ERK_N, Length: 1078, dtype: int64

NaN         3
1.566568    1
1.288566    1
0.775422    1
1.254325    1
           ..
1.187508    1
1.138284    1
0.956864    1
1.253606    1
0.984375    1
Name: GSK3B_N, Length: 1078, dtype: int64

NaN         3
0.274420    1
0.265823    1
0.178050    1
0.253696    1
           ..
0.240480    1
```

```
0.179443    1
0.227548    1
0.239894    1
0.308594    1
Name: JNK_N, Length: 1078, dtype: int64

NaN         7
0.275229    2
0.270520    1
0.241012    1
0.299435    1
           ..
0.217975    1
0.287178    1
0.276433    1
0.282486    1
0.310592    1
Name: MEK_N, Length: 1073, dtype: int64

NaN         3
0.666667    2
0.776860    2
0.620888    1
0.739193    1
           ..
0.778749    1
0.773109    1
0.764164    1
0.672022    1
0.898438    1
Name: TRKA_N, Length: 1076, dtype: int64

NaN         3
0.162162    2
0.195556    2
0.187737    2
0.150356    1
           ..
0.169674    1
0.255417    1
0.168565    1
0.175960    1
0.187635    1
Name: RSK_N, Length: 1075, dtype: int64

NaN         3
0.435954    1
0.488346    1
0.437253    1
0.421505    1
           ..
0.391051    1
0.458036    1
0.403786    1
0.367804    1
0.426378    1
Name: APP_N, Length: 1078, dtype: int64

NaN         18
2.316108     1
1.989009     1
```

```
2.435460     1
2.170877     1
            ..
3.157139     1
1.791115     1
3.194291     1
2.307475     1
1.889974     1
Name: Bcatenin_N, Length: 1063, dtype: int64

NaN         3
0.338881    1
0.415011    1
0.323922    1
0.346848    1
           ..
0.934453    1
0.624297    1
0.344384    1
1.862032    1
0.304688    1
Name: SOD1_N, Length: 1078, dtype: int64

NaN         3
0.484417    1
0.440535    1
0.335592    1
0.473757    1
           ..
0.444682    1
0.373093    1
0.507546    1
0.375068    1
0.459153    1
Name: MTOR_N, Length: 1078, dtype: int64

NaN         3
0.333333    2
0.472924    2
0.501973    1
0.767414    1
           ..
0.260482    1
0.388685    1
0.337080    1
0.409194    1
0.375000    1
Name: P38_N, Length: 1076, dtype: int64

NaN         3
0.664087    1
0.821397    1
0.647831    1
0.660803    1
           ..
0.980997    1
0.868649    1
0.779984    1
0.669200    1
0.671875    1
Name: pMTOR_N, Length: 1078, dtype: int64
```

```
NaN         3
0.568279    1
0.732805    1
0.639824    1
0.597499    1
           ..
0.588816    1
0.654792    1
0.679691    1
0.614897    1
0.500000    1
Name: DSCR1_N, Length: 1078, dtype: int64

NaN         3
0.333333    2
0.367718    2
0.397461    1
0.347080    1
           ..
0.356382    1
0.430054    1
0.407770    1
0.353162    1
0.375000    1
Name: AMPKA_N, Length: 1076, dtype: int64

NaN         3
0.658220    1
0.564627    1
0.477753    1
0.449673    1
           ..
0.560265    1
0.551375    1
0.536162    1
0.593223    1
0.500000    1
Name: NR2B_N, Length: 1078, dtype: int64

NaN         3
0.346698    1
0.269376    1
0.354723    1
0.482060    1
           ..
0.343494    1
0.631052    1
0.396394    1
0.411307    1
0.312500    1
Name: pNUMB_N, Length: 1078, dtype: int64

NaN         3
0.310568    1
0.279404    1
0.290558    1
0.273450    1
           ..
0.256496    1
0.267165    1
```

```
0.356240    1
0.324729    1
0.306349    1
Name: RAPTOR_N, Length: 1078, dtype: int64

NaN         3
0.375000    2
0.315566    2
0.483975    1
0.395578    1
           ..
0.478614    1
0.502497    1
0.379032    1
0.391859    1
0.313313    1
Name: TIAM1_N, Length: 1076, dtype: int64

NaN         3
0.500000    2
0.457021    1
0.146744    1
0.460311    1
           ..
0.172909    1
0.418000    1
0.204570    1
0.352199    1
0.250000    1
Name: pP70S6_N, Length: 1077, dtype: int64

0.170561    1
0.172089    1
0.182155    1
0.180839    1
0.191481    1
           ..
0.177308    1
0.191941    1
0.136392    1
0.180791    1
0.211379    1
Name: NUMB_N, Length: 1080, dtype: int64

0.949038    1
1.005561    1
0.880478    1
0.815421    1
0.955463    1
           ..
1.160493    1
0.808441    1
0.621137    1
1.034466    1
0.795393    1
Name: P70S6_N, Length: 1080, dtype: int64

0.136234    1
0.187305    1
0.150588    1
0.176245    1
```

```
0.154845    1
           ..
0.177724    1
0.129624    1
0.155770    1
0.164132    1
0.138672    1
Name: pGSK3B_N, Length: 1080, dtype: int64

2.107487    1
1.377177    1
0.680375    1
2.297259    1
1.307285    1
           ..
2.519636    1
2.111795    1
2.636905    1
1.221793    1
1.015625    1
Name: pPKCG_N, Length: 1080, dtype: int64

0.297832    1
0.296240    1
0.285391    1
0.271077    1
0.293819    1
           ..
0.310830    1
0.302796    1
0.310498    1
0.355993    1
0.290466    1
Name: CDK5_N, Length: 1080, dtype: int64

0.689591    1
0.289984    1
0.297363    1
0.375343    1
0.584942    1
           ..
0.371497    1
0.491090    1
0.538739    1
0.485493    1
0.250000    1
Name: S6_N, Length: 1080, dtype: int64

1.214885    1
1.430719    1
1.237679    1
1.414077    1
1.598518    1
           ..
1.317706    1
1.211773    1
1.217917    1
0.921927    1
1.056348    1
Name: ADARB1_N, Length: 1080, dtype: int64
```

```
0.087164    1
0.190230    1
0.306249    1
0.141747    1
0.198303    1
           ..
0.172076    1
0.142609    1
0.102433    1
0.184482    1
0.313857    1
Name: AcetylH3K9_N, Length: 1080, dtype: int64

0.178487    1
0.185960    1
0.177636    1
0.167565    1
0.189965    1
           ..
0.166808    1
0.221612    1
0.175895    1
0.153285    1
0.146484    1

Name: RRP1_N, Length: 1080, dtype: int64

0.178234    1
0.171822    1
0.161919    1
0.183004    1
0.144771    1
           ..
0.163157    1
0.182778    1
0.179943    1
0.184477    1
0.188697    1
Name: BAX_N, Length: 1080, dtype: int64

0.123284    1
0.117778    1
0.136584    1
0.115571    1
0.137112    1
           ..
0.109865    1
0.113552    1
0.151461    1
0.139718    1
0.108303    1
Name: ARC_N, Length: 1080, dtype: int64

0.155331    2
0.136794    1
0.140167    1
0.175346    1
0.155779    1
           ..
0.131840    1
0.149030    1
```

```
0.143520    1
0.117226    1
0.157760    1
Name: ERBB4_N, Length: 1079, dtype: int64

0.173913    2
0.191373    1
0.184705    1
0.192618    1
0.191585    1
           ..
0.185596    1
0.170476    1
0.204250    1
0.166173    1
0.206560    1
Name: nNOS_N, Length: 1079, dtype: int64

0.297862    1
0.152297    1
0.257896    1
0.139979    1
0.224861    1
           ..
0.202646    1
0.255569    1
0.143535    1
0.189121    1
0.164479    1
Name: Tau_N, Length: 1080, dtype: int64

0.130435    2
0.098555    1
0.122602    1
0.120635    1
0.138530    1
           ..
0.116837    1
0.117500    1
0.127442    1
0.120356    1
0.132812    1
Name: GFAP_N, Length: 1079, dtype: int64

0.189949    1
0.221600    1
0.246252    1
0.189687    1
0.168231    1
           ..
0.215253    1
0.294030    1
0.217799    1
0.174588    1
0.194739    1
Name: GluR3_N, Length: 1080, dtype: int64

0.110996    2
0.099600    1
0.122228    1
0.102332    1
```

```
0.155764    1
           ..
0.145065    1
0.146239    1
0.121123    1
0.155283    1
0.105126    1
Name: GluR4_N, Length: 1079, dtype: int64

0.503901    1
0.460272    1
0.623366    1
0.678803    1
0.553856    1
           ..
0.466362    1
0.562792    1
0.525681    1
0.588496    1
0.623047    1
Name: IL1B_N, Length: 1080, dtype: int64

0.308899    1
0.304809    1
0.281441    1
0.317196    1
0.256287    1
           ..
0.263058    1
0.270881    1
0.295218    1
0.264688    1
0.307169    1
Name: P3525_N, Length: 1080, dtype: int64

1.592678    1
1.667761    1
1.581123    1
1.657349    1
1.621090    1
           ..
1.721678    1
1.185995    1
1.413082    1
1.746246    1
1.812500    1
Name: pCASP9_N, Length: 1080, dtype: int64

2.531140    1
2.543164    1
1.809995    1
2.174248    1
2.026886    1
           ..
2.241833    1
1.818907    1
2.329648    1
1.612601    1
2.500000    1
Name: PSD95_N, Length: 1080, dtype: int64
```

```
0.171454    2
0.192129    1
0.164642    1
0.155340    1
0.139986    1
           ..
0.176397    1
0.195418    1
0.150296    1
0.197839    1
0.172229    1
Name: SNCA_N, Length: 1079, dtype: int64

1.269499    1
1.481566    1
1.146432    1
1.329207    1
1.169842    1
           ..
1.086489    1
1.419114    1
0.995907    1
1.221415    1
1.500000    1
Name: Ubiquitin_N, Length: 1080, dtype: int64

0.780729    1
0.718491    1
0.827264    1
1.150855    1
0.826923    1
           ..
0.802191    1
0.815956    1
0.941176    1
0.847205    1
1.000000    1
Name: pGSK3B_Tyr216_N, Length: 1080, dtype: int64

0.217774    1
0.271527    1
0.281851    1
0.219992    1
0.173505    1
           ..
0.228705    1
0.223233    1
0.229276    1
0.178426    1
0.250000    1
Name: SHH_N, Length: 1080, dtype: int64

NaN         213
0.200276      2
0.158962      1
0.127629      1
0.154193      1
           ...
0.161236      1
0.130649      1
0.146644      1
```

```
0.202864      1
0.160749      1
Name: BAD_N, Length: 867, dtype: int64

NaN         285
0.098062      1
0.134224      1
0.104274      1
0.125965      1
           ...
0.170222      1
0.140527      1
0.114969      1
0.156098      1
0.149341      1
Name: BCL2_N, Length: 796, dtype: int64

0.123284    1
0.117778    1
0.136584    1
0.115571    1
0.137112    1
           ..
0.109865    1
0.113552    1
0.151461    1
0.139718    1
0.108303    1
Name: pS6_N, Length: 1080, dtype: int64

NaN         75
0.120234     1
0.148578     1
0.110486     1
0.106987     1
            ..
0.104697     1
0.103836     1
0.118575     1
0.129548     1
0.094227     1
Name: pCFOS_N, Length: 1006, dtype: int64

0.487991    2
0.474465    1
0.513971    1
0.381729    1
0.457260    1
           ..
0.518070    1
0.486912    1
0.506980    1
0.554550    1
0.406250    1
Name: SYP_N, Length: 1079, dtype: int64

NaN         180
0.102054      1
0.206347      1
0.159205      1
0.116050      1
```

```
           ...
0.123357      1
0.209641      1
0.107223      1
0.114257      1
0.152119      1
Name: H3AcK18_N, Length: 901, dtype: int64

NaN         210
0.148252      1
0.145337      1
0.192383      1
0.153314      1
           ...
0.176071      1
0.143156      1
0.185249      1
0.282808      1
0.179941      1
Name: EGR1_N, Length: 871, dtype: int64

NaN         270
0.237160      1
0.182055      1
0.214611      1
0.155251      1
           ...
0.208481      1
0.276836      1
0.206853      1
0.199104      1
0.150355      1
Name: H3MeK4_N, Length: 811, dtype: int64

1.213191    1
1.266245    1
1.084657    1
0.854759    1
1.199038    1
           ..
1.058489    1
1.857424    1
1.899171    1
1.696719    1
1.000000    1
Name: CaNA_N, Length: 1080, dtype: int64

Control    570
Ts65Dn     510
Name: Genotype, dtype: int64

Memantine    570
Saline       510
Name: Treatment, dtype: int64

S/C    555
C/S    525
Name: Behavior, dtype: int64

c-SC-m    150
c-CS-m    150
```

```
c-CS-s    135
t-CS-m    135
t-SC-s    135
t-SC-m    135
c-SC-s    135
t-CS-s    105
Name: class, dtype: int64
```

In [7]:

```
# Changing datatypes
# Convert the Genotype variable into Category type
mice.loc[: , 'Genotype'] = mice.loc[: , 'Genotype'].astype('category')
mice['Genotype'].dtype
```

Out[7]:

```
CategoricalDtype(categories=['Control', 'Ts65Dn'], ordered=False)
```

In [8]:

```
# Convert the Treatment variable into Category type
mice.loc[: , 'Treatment'] = mice.loc[: , 'Treatment'].astype('category')
mice['Treatment'].dtype
```

Out[8]:

```
CategoricalDtype(categories=['Memantine', 'Saline'], ordered=False)
```

In [9]:

```
# Convert the Behavior variable into Category type
mice.loc[: , 'Behavior'] = mice.loc[: , 'Behavior'].astype('category')
mice['Behavior'].dtype
```

Out[9]:

```
CategoricalDtype(categories=['C/S', 'S/C'], ordered=False)
```

In [10]:

```
# Convert the class variable into Category type
mice.loc[: , 'class'] = mice.loc[: , 'class'].astype('category')
mice['class'].dtype
```

Out[10]:

```
CategoricalDtype(categories=['c-CS-m', 'c-CS-s', 'c-SC-m', 'c-SC-s', 't-CS-
m', 't-CS-s',
                  't-SC-m', 't-SC-s'],
                 ordered=False)
```

In [12]:

```
# To check how NaN values and their previous/next values related to the categorical values
# Found out the categorical values are in sequence. As a result, we can change the NaN valu
# But, some of the NaN values in a sequence will have the same propogated value if its ffil
# So, null values will be replaced by interpolation method within each category by indices
# If after interpolation, the dataset cotains nulls values then those remaining would be re
mice.loc[28:, 'H3MeK4_N':'class'].head(40)
```

Out[12]:

	H3MeK4_N	CaNA_N	Genotype	Treatment	Behavior	class
28	0.217621	0.994404	Control	Memantine	C/S	c-CS-m
29	0.208286	1.030948	Control	Memantine	C/S	c-CS-m
30	NaN	1.649983	Control	Memantine	C/S	c-CS-m
31	NaN	1.638988	Control	Memantine	C/S	c-CS-m
32	NaN	1.564925	Control	Memantine	C/S	c-CS-m
33	NaN	1.682222	Control	Memantine	C/S	c-CS-m
34	NaN	1.660352	Control	Memantine	C/S	c-CS-m
35	NaN	1.610816	Control	Memantine	C/S	c-CS-m
36	NaN	1.727381	Control	Memantine	C/S	c-CS-m
37	NaN	1.650846	Control	Memantine	C/S	c-CS-m

In [13]:

```
# Using random selection from unique list method
#for col in mice.columns:
    #sample_list = mice[mice[col].notnull()][col].unique()
    #mice[col] = mice.apply(lambda r: random.choice(sample_list) if pd.isnull(r[col]) else

# Using mean method
# for col in mice.columns[1:78]:
    #mice[col].fillna(mice[col].mean(), axis = 0, inplace = True)

# 1. Interpolation method
mice1 = mice.copy()
mice1= mice1.groupby(['Genotype', 'Behavior', 'Treatment', 'class']).apply(lambda group: gr
mice1.reset_index(drop =  True, inplace = True)
mice.loc[:, 'MouseID':'CaNA_N'] = mice1.values

# 2. Using bfill method
for i in range(1,79):
    col = mice.columns[i]
    mice.loc[: , mice.columns[i]].fillna(method = 'bfill', inplace =True)
```

In [14]:

```
mice.isna().sum().sum()
```

Out[14]:

```
0
```

In [15]:

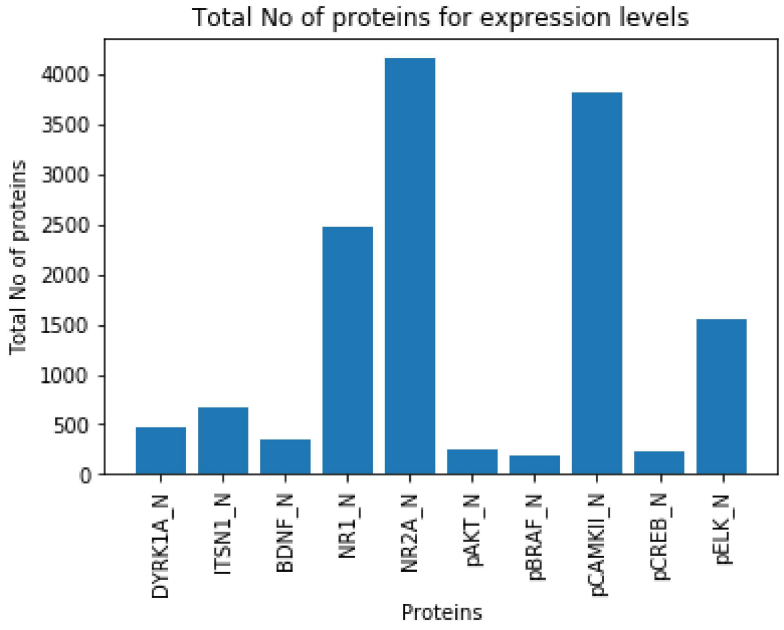
```
mice.loc[30:, 'H3MeK4_N':'class'].head(15)
```

Out[15]:

	H3MeK4_N	CaNA_N	Genotype	Treatment	Behavior	class
30	0.206878	1.649983	Control	Memantine	C/S	c-CS-m
31	0.205470	1.638988	Control	Memantine	C/S	c-CS-m
32	0.204062	1.564925	Control	Memantine	C/S	c-CS-m
33	0.202653	1.682222	Control	Memantine	C/S	c-CS-m
34	0.201245	1.660352	Control	Memantine	C/S	c-CS-m
35	0.199837	1.610816	Control	Memantine	C/S	c-CS-m
36	0.198429	1.727381	Control	Memantine	C/S	c-CS-m
37	0.197021	1.650846	Control	Memantine	C/S	c-CS-m
38	0.195613	1.555327	Control	Memantine	C/S	c-CS-m
39	0.194205	1.655576	Control	Memantine	C/S	c-CS-m
40	0.192797	1.681703	Control	Memantine	C/S	c-CS-m
41	0.191389	1.585793	Control	Memantine	C/S	c-CS-m
42	0.189981	1.615576	Control	Memantine	C/S	c-CS-m
43	0.188573	1.561914	Control	Memantine	C/S	c-CS-m
44	0.187165	1.530458	Control	Memantine	C/S	c-CS-m

In [16]:

```
# Task 2.1: Explore the columns
avg = mice[mice.columns[1:11]].sum()
index = mice.columns[1:11]
plt.bar(index, avg)
plt.xlabel('Proteins')
plt.xticks(rotation=90)
plt.ylabel('Total No of proteins')
plt.title('Total No of proteins for expression levels')
plt.show()
```



In [17]:

```
mice.describe()
```

Out[17]:

	DYRK1A_N	ITSN1_N	BDNF_N	NR1_N	NR2A_N	pAKT_N	pBRAI
count	1080.000000	1080.000000	1080.000000	1080.000000	1080.000000	1080.000000	1080.000
mean	0.425762	0.617213	0.319154	2.297361	3.843463	0.233238	0.181
std	0.249018	0.251302	0.049338	0.346862	0.931974	0.041598	0.027
min	0.145327	0.245359	0.115181	1.330831	1.737540	0.063236	0.064
25%	0.288163	0.473669	0.287650	2.059152	3.160287	0.205821	0.164
50%	0.366540	0.566365	0.316703	2.297299	3.749885	0.231246	0.182
75%	0.487574	0.697500	0.348235	2.528035	4.425107	0.257450	0.197
max	2.516367	2.602662	0.497160	3.757641	8.482553	0.539050	0.317

8 rows × 77 columns

In [18]:

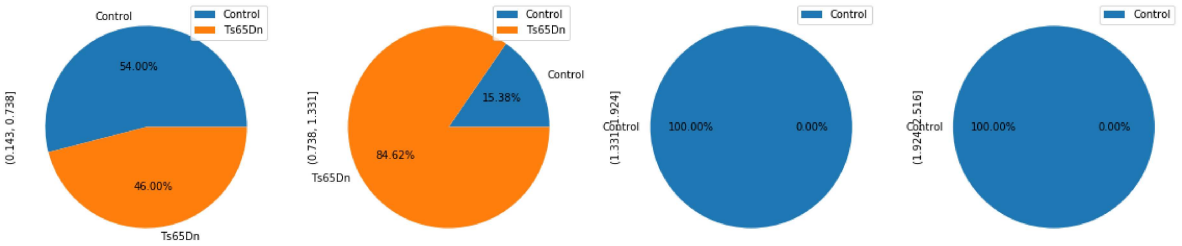
```
mice.groupby(by='Genotype')['DYRK1A_N'].describe()
```

Out[18]:

Genotype	count	mean	std	min	25%	50%	75%	max
Control	570.0	0.404756	0.300553	0.145327	0.270523	0.328347	0.440944	2.516367
Ts65Dn	510.0	0.449238	0.171538	0.163325	0.329478	0.408165	0.527983	0.992220

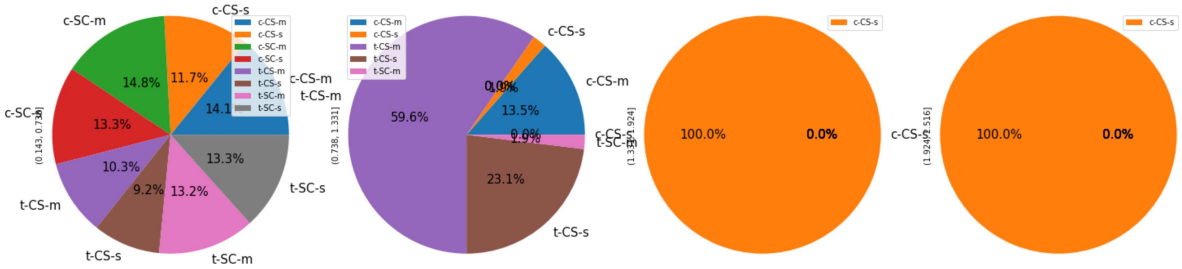
In [19]:

```
# Task 2.2: Relationships between columns
mice['DYRK1A_N'] = pd.cut(mice['DYRK1A_N'], bins=4)
grouped = mice.groupby(['Genotype', 'DYRK1A_N'])['Genotype'].size()
grouped.unstack().plot.pie(subplots=True, figsize=(20, 20), autopct='%.2f%%')
plt.show()
```



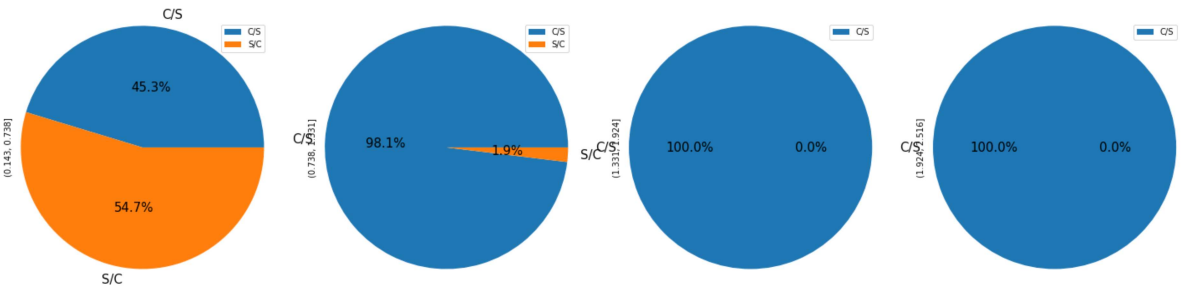
In [20]:

```
grouped = mice.groupby(['class', 'DYRK1A_N'])['class'].size()
grouped.unstack().plot.pie(subplots=True, radius=1.2, figsize=(25, 20), fontsize = 15, pctc
plt.show()
```



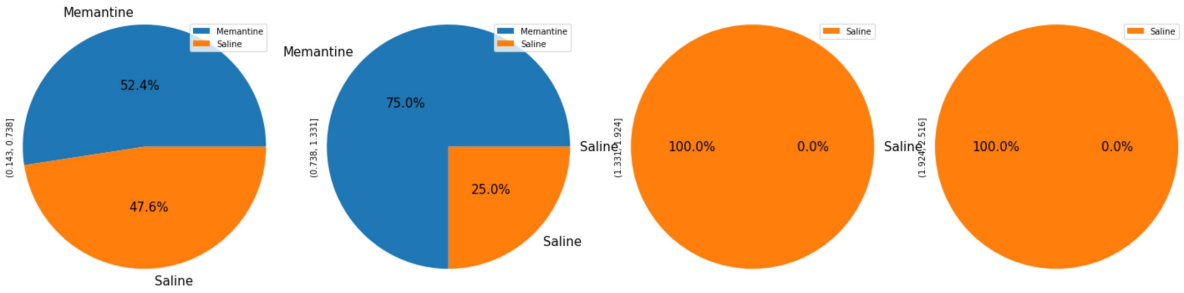
In [21]:

```
grouped = mice.groupby(['Behavior', 'DYRK1A_N'])['Behavior'].size()
grouped.unstack().plot.pie(subplots=True, radius=1.2, figsize=(25, 20), fontsize = 15, pctc
plt.show()
```



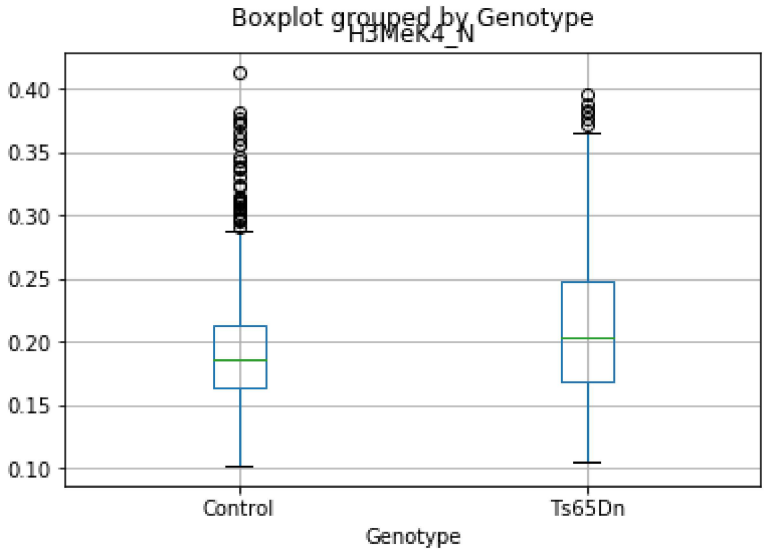
In [22]:

```
grouped = mice.groupby(['Treatment', 'DYRK1A_N'])['Treatment'].size()
grouped.unstack().plot.pie(subplots=True, radius=1.2, figsize=(25, 20), fontsize = 15, pctc
plt.show()
```



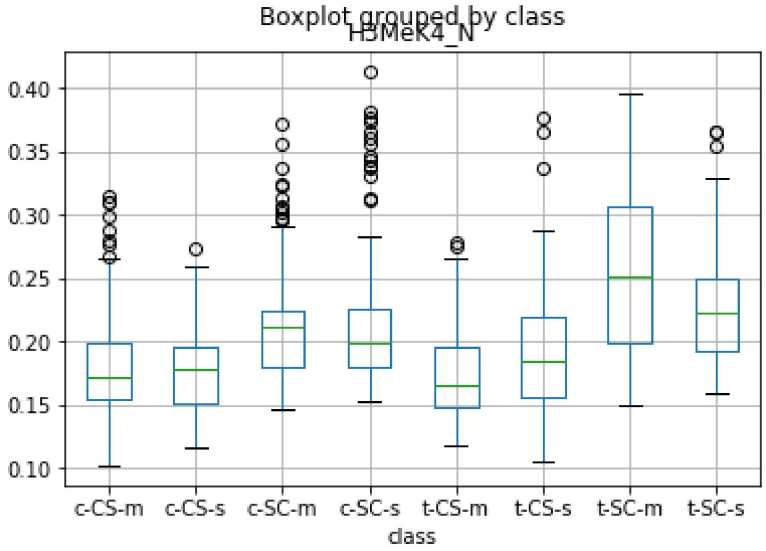
In [23]:

```
mice.boxplot(column = 'H3MeK4_N', by = 'Genotype')
plt.show()
```



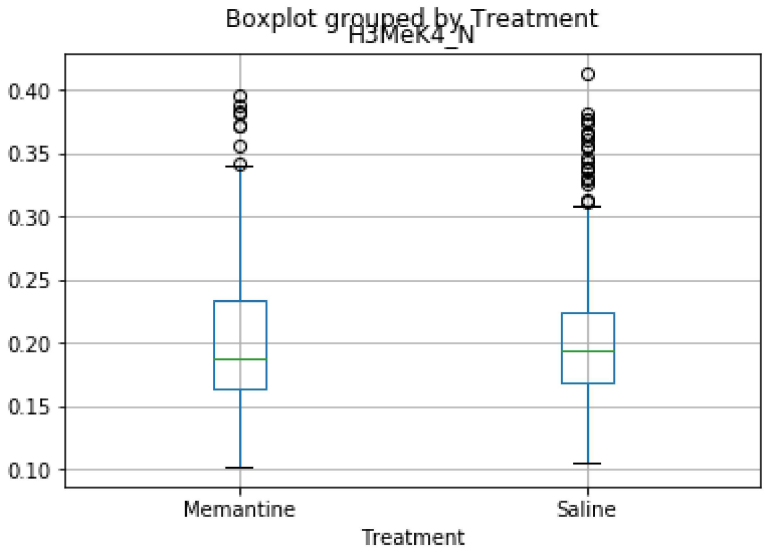
In [24]:

```
mice.boxplot(column = 'H3MeK4_N', by = 'class')
plt.show()
```



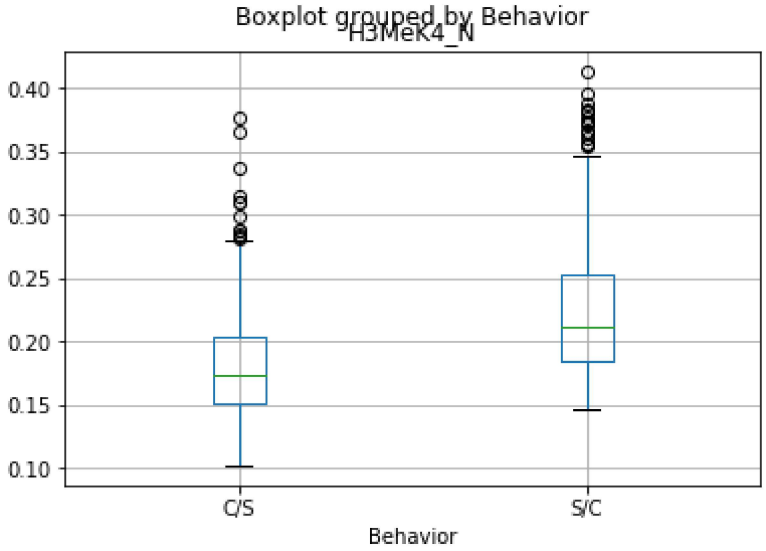
In [25]:

```
mice.boxplot(column = 'H3MeK4_N', by = 'Treatment')
plt.show()
```



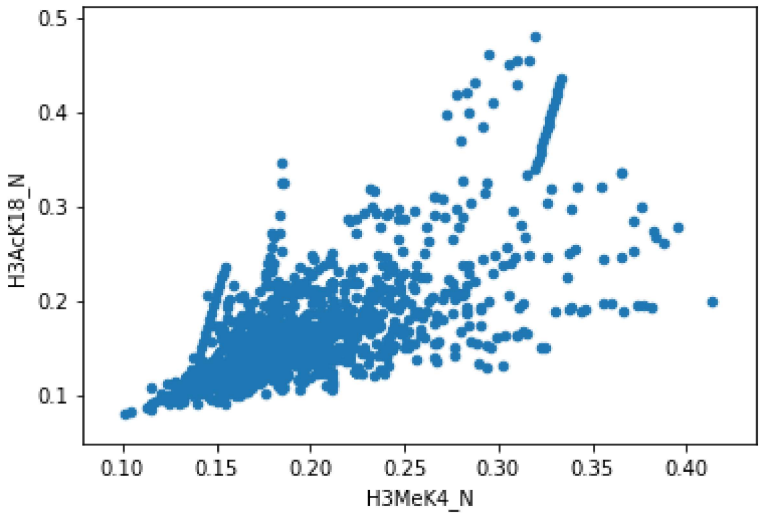
In [26]:

```
mice.boxplot(column = 'H3MeK4_N', by = 'Behavior')
plt.show()
```



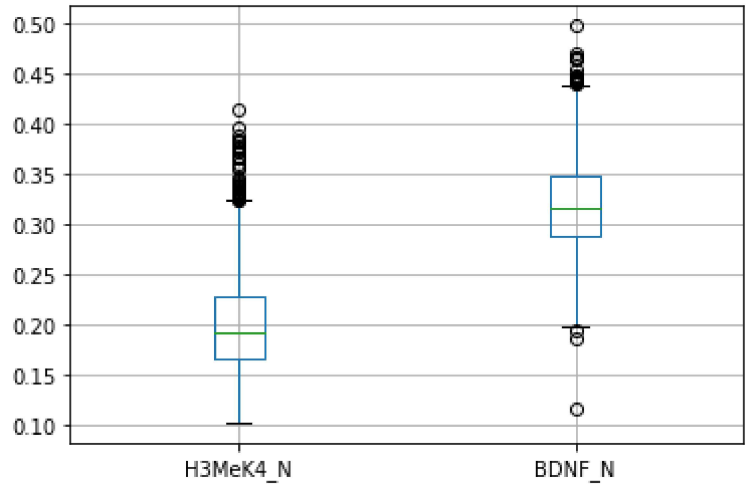
In [27]:

```
mice.plot.scatter(x='H3MeK4_N', y='H3AcK18_N')
plt.show()
```



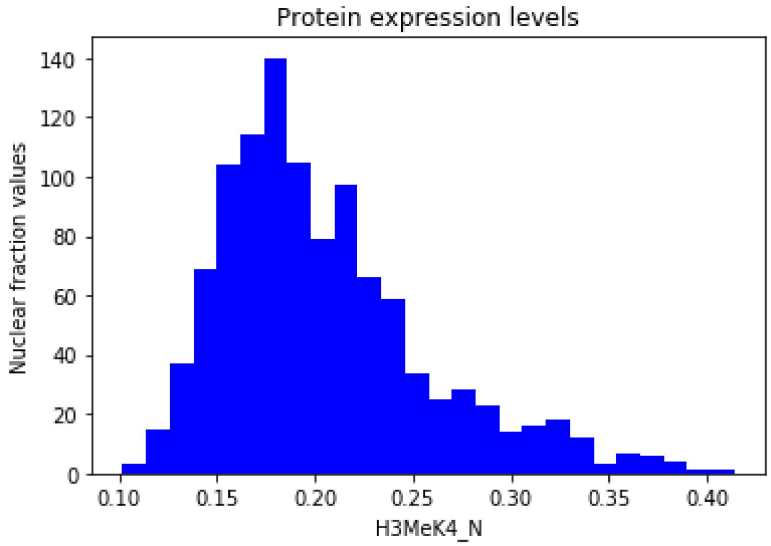
In [28]:

```
mice.boxplot(column=['H3MeK4_N', 'BDNF_N'])
plt.show()
```



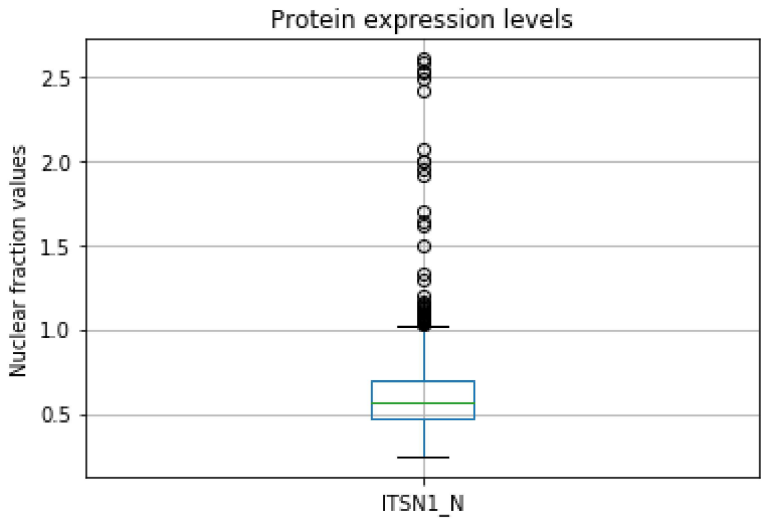
In [29]:

```
plt.hist(mice['H3MeK4_N'], color = 'blue', bins = 'auto')
plt.xlabel('H3MeK4_N')
plt.ylabel('Nuclear fraction values')
plt.title('Protein expression levels')
plt.show()
```



In [30]:

```
for i in mice.columns[2:12]:
    mice.boxplot(column=i)
    plt.ylabel('Nuclear fraction values')
    plt.title('Protein expression levels')
    plt.show()
```



In [31]:

```
mice['DYRK1A_N'] = mice1['DYRK1A_N'].values
```

In [32]:

```
# Data Modelling
X_data = mice.loc[:, 'DYRK1A_N':'CaNA_N']
X_data
```

Out[32]:

	DYRK1A_N	ITSN1_N	BDNF_N	NR1_N	NR2A_N	pAKT_N	pBRAF_N	pCAMKII_N	pC
0	0.503644	0.747193	0.430175	2.816329	5.990152	0.218830	0.177565	2.373744	0
1	0.514617	0.689064	0.411770	2.789514	5.685038	0.211636	0.172817	2.292150	0
2	0.509183	0.730247	0.418309	2.687201	5.622059	0.209011	0.175722	2.283337	0
3	0.442107	0.617076	0.358626	2.466947	4.979503	0.222886	0.176463	2.152301	0
4	0.434940	0.617430	0.358802	2.365785	4.718679	0.213106	0.173627	2.134014	0
...	...	...	...	...	...	...	...	...	
1075	0.254860	0.463591	0.254860	2.092082	2.600035	0.211736	0.171262	2.483740	0
1076	0.272198	0.474163	0.251638	2.161390	2.801492	0.251274	0.182496	2.512737	0
1077	0.228700	0.395179	0.234118	1.733184	2.220852	0.220665	0.161435	1.989723	0
1078	0.221242	0.412894	0.243974	1.876347	2.384088	0.208897	0.173623	2.086028	0
1079	0.302626	0.461059	0.256564	2.092790	2.594348	0.251001	0.191811	2.361816	0

1080 rows × 77 columns

In [33]:

```
y_data = mice['class']
y_data
```

Out[33]:

```
0       c-CS-m
1       c-CS-m
2       c-CS-m
3       c-CS-m
4       c-CS-m
         ...
1075    t-SC-s
1076    t-SC-s
1077    t-SC-s
1078    t-SC-s
1079    t-SC-s
Name: class, Length: 1080, dtype: category
Categories (8, object): [c-CS-m, c-CS-s, c-SC-m, c-SC-s, t-CS-m, t-CS-s, t-S
C-m, t-SC-s]
```

In [34]:

```
# 30 % for testing will give a better indication of the models performance on unseen data.
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_data, y_data, test_size = .3, random_
print(X_train.shape)
print(X_train[:5])
```

```
(756, 77)
     DYRK1A_N   ITSN1_N    BDNF_N     NR1_N    NR2A_N    pAKT_N   pBRAF_N  \
62   0.395358  0.576313  0.342568  2.356046  5.326220  0.207413  0.168601
863  0.617112  0.687791  0.292148  2.113193  3.393251  0.237013  0.176033
780  0.333636  0.602178  0.324040  2.834414  5.132132  0.242998  0.179461
291  0.266736  0.445252  0.281439  2.186127  3.363951  0.237156  0.181111
845  0.474593  0.702149  0.326170  2.174734  3.998393  0.276762  0.198634

     pCAMKII_N   pCREB_N    pELK_N  ...      SHH_N     BAD_N    BCL2_N  \
62    3.511544  0.181580  1.416199  ...   0.238161  0.156498  0.133049
863   4.108277  0.232895  1.221071  ...   0.223820  0.160849  0.134713
780   6.358402  0.216286  1.278268  ...   0.223717  0.134261  0.120728
291   4.167099  0.198236  1.179381  ...   0.184071  0.155141  0.149856
845   2.705965  0.198835  1.695120  ...   0.206222  0.187507  0.149720

        pS6_N   pCFOS_N     SYP_N  H3AcK18_N    EGR1_N  H3MeK4_N    CaNA_N
62   0.134710  0.122870  0.493719   0.170838  0.254454  0.161821  1.723145
863  0.111735  0.114892  0.487721   0.187862  0.173215  0.220400  1.760042
780  0.135943  0.108342  0.450570   0.115987  0.154446  0.169126  1.062237
291  0.128170  0.127478  0.489281   0.218996  0.208506  0.211235  1.163785
845  0.123137  0.109931  0.404697   0.176898  0.152361  0.215878  1.228156

[5 rows x 77 columns]
```

In [35]:

```
print(y_train.shape)
print(y_train[:5])
```

```
(756,)
62     c-CS-m
863    t-CS-s
780    t-SC-m
291    c-SC-m
845    t-CS-s
Name: class, dtype: category
Categories (8, object): [c-CS-m, c-CS-s, c-SC-m, c-SC-s, t-CS-m, t-CS-s, t-S
C-m, t-SC-s]
```

In [36]:

```
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score

# creating odd list of K for KNN
neighbors = list(range(1, 9, 2))

# perform 10-fold cross validation
for k in neighbors:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, X_train, y_train, cv=10, scoring='accuracy')
    print("K score for " +str(k)+ " is " +str(scores.mean()))
    print()
```

```
K score for 1 is 0.9853649248079627

K score for 3 is 0.9392366071036957

K score for 5 is 0.9127893365614884

K score for 7 is 0.8944105546637193
```

In [37]:

```
# K=1 could result in 100% accuracy or overfitting
# Instead, k=3 would be the value for KNeighbors
clf = KNeighborsClassifier(3)
fit = clf.fit(X_train, y_train)
y_pre = fit.predict(X_test)
```

In [38]:

```
metrics.accuracy_score(y_test, y_pre)
```

Out[38]:

```
0.9351851851851852
```

In [39]:

```
cm = confusion_matrix(y_test, y_pre)
cm
```

Out[39]:

```
array([[33,  0,  0,  0,  0,  0,  0,  0],
       [ 2, 34,  0,  0,  1,  4,  0,  0],
       [ 0,  0, 48,  0,  0,  0,  4,  0],
       [ 0,  0,  0, 47,  0,  0,  0,  0],
       [ 2,  0,  0,  0, 32,  0,  0,  1],
       [ 2,  0,  0,  0,  0, 29,  0,  0],
       [ 0,  0,  2,  0,  0,  0, 39,  0],
       [ 0,  0,  2,  0,  0,  0,  1, 41]], dtype=int64)
```

In [40]:

```
# parameter tuning
# stratify base on class variable ie equal category distributions in train and test split a
X_train, X_test, y_train, y_test = train_test_split(X_data, y_data, test_size = .3, random_
clf = KNeighborsClassifier(3)
fit = clf.fit(X_train, y_train)
y_pre = fit.predict(X_test)
metrics.accuracy_score(y_test, y_pre)
```

Out[40]:

```
0.9506172839506173
```

In [41]:

```
print(y_train[:5])
```

```
246    c-SC-m
611    t-CS-m
335    c-CS-s
203    c-SC-m
253    c-SC-m
Name: class, dtype: category
Categories (8, object): [c-CS-m, c-CS-s, c-SC-m, c-SC-s, t-CS-m, t-CS-s, t-S
C-m, t-SC-s]
```

In [42]:

```
# Parameters with default value gived the best accuracy score for unequal and equal distrib
from sklearn.tree import DecisionTreeClassifier
X_train, X_test, y_train, y_test = train_test_split(X_data, y_data, test_size = .3, random_
clf = DecisionTreeClassifier(random_state = 4)
fit = clf.fit(X_train, y_train)
y_pre = fit.predict(X_test)
metrics.accuracy_score(y_test, y_pre)
```

Out[42]:

```
0.8549382716049383
```

In [43]:

```
cm = confusion_matrix(y_test, y_pre)
cm
```

Out[43]:

```
array([[30,  2,  0,  0,  1,  0,  0,  0],
       [ 7, 30,  0,  0,  1,  3,  0,  0],
       [ 1,  0, 43,  1,  0,  0,  6,  1],
       [ 0,  0,  5, 41,  0,  0,  0,  1],
       [ 0,  4,  0,  0, 27,  4,  0,  0],
       [ 0,  1,  0,  0,  3, 27,  0,  0],
       [ 0,  1,  2,  1,  0,  0, 36,  1],
       [ 0,  0,  1,  0,  0,  0,  0, 43]], dtype=int64)
```

In [44]:

```
# parameter tuning
# stratify base on class variable ie equal category distributions in train and test split a
X_train, X_test, y_train, y_test = train_test_split(X_data, y_data, test_size = .3, random_
clf = DecisionTreeClassifier(random_state = 4)
fit = clf.fit(X_train, y_train)
y_pre = fit.predict(X_test)
metrics.accuracy_score(y_test, y_pre)
```

Out[44]:

```
0.8703703703703703
```

In [ ]:

```

```

In [ ]:

```

```