In [33]:
```
import pandas as pd
import numpy as np
from sklearn import linear_model
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
```

In [34]:
```
data=pd.read_csv(r'E:\Downloads\AI-Data.csv')
data.head()
```

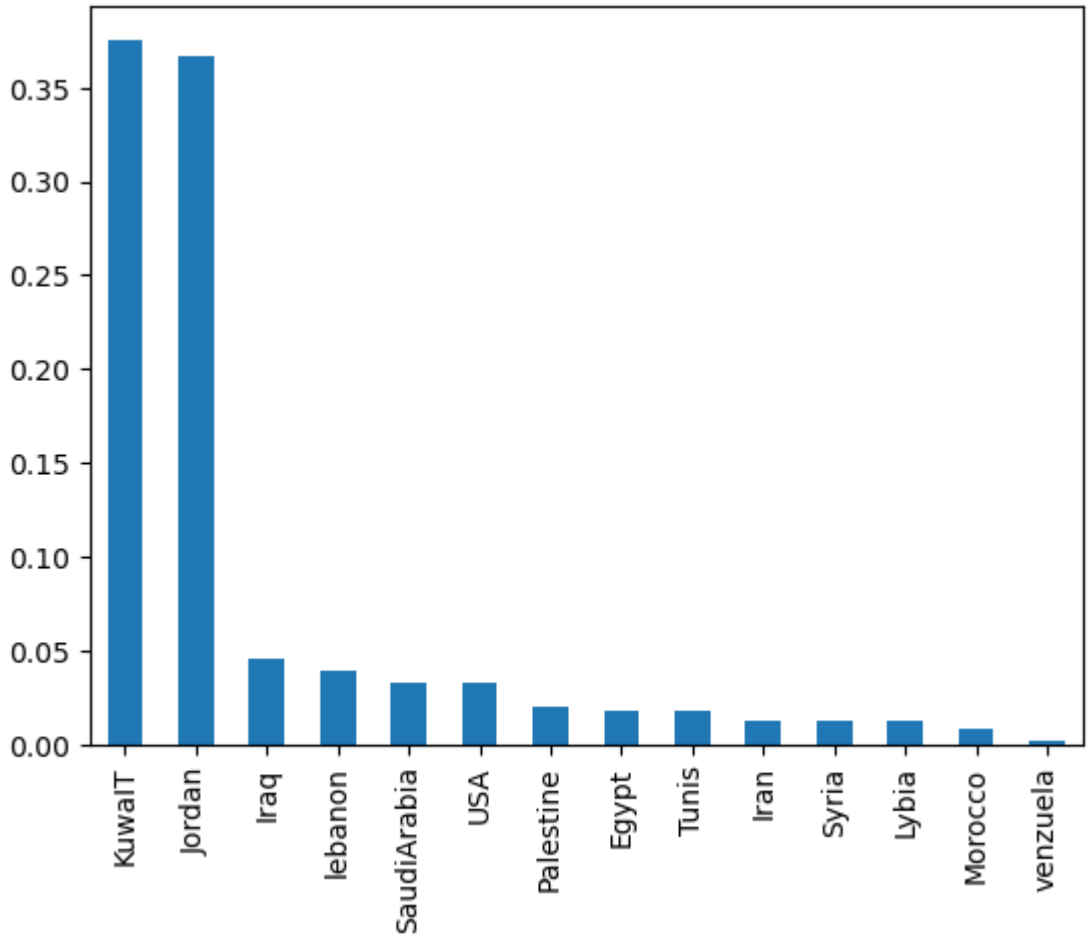
Out[34]:

	gender	NationalITy	PlaceofBirth	StageID	GradeID	SectionID	Topic	Semester	Relation
0	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father
1	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father
2	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father
3	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father
4	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father

In [35]:
```
data['PlaceofBirth'].value_counts()
print('Percentage',data.PlaceofBirth.value_counts(normalize=True))
data.PlaceofBirth.value_counts(normalize=True).plot(kind='bar')
```

```
Percentage KuwaIT         0.375000
Jordan          0.366667
Iraq            0.045833
lebanon         0.039583
SaudiArabia     0.033333
USA             0.033333
Palestine       0.020833
Egypt           0.018750
Tunis           0.018750
Iran            0.012500
Syria           0.012500
Lybia           0.012500
Morocco         0.008333
venzuela        0.002083
Name: PlaceofBirth, dtype: float64
```

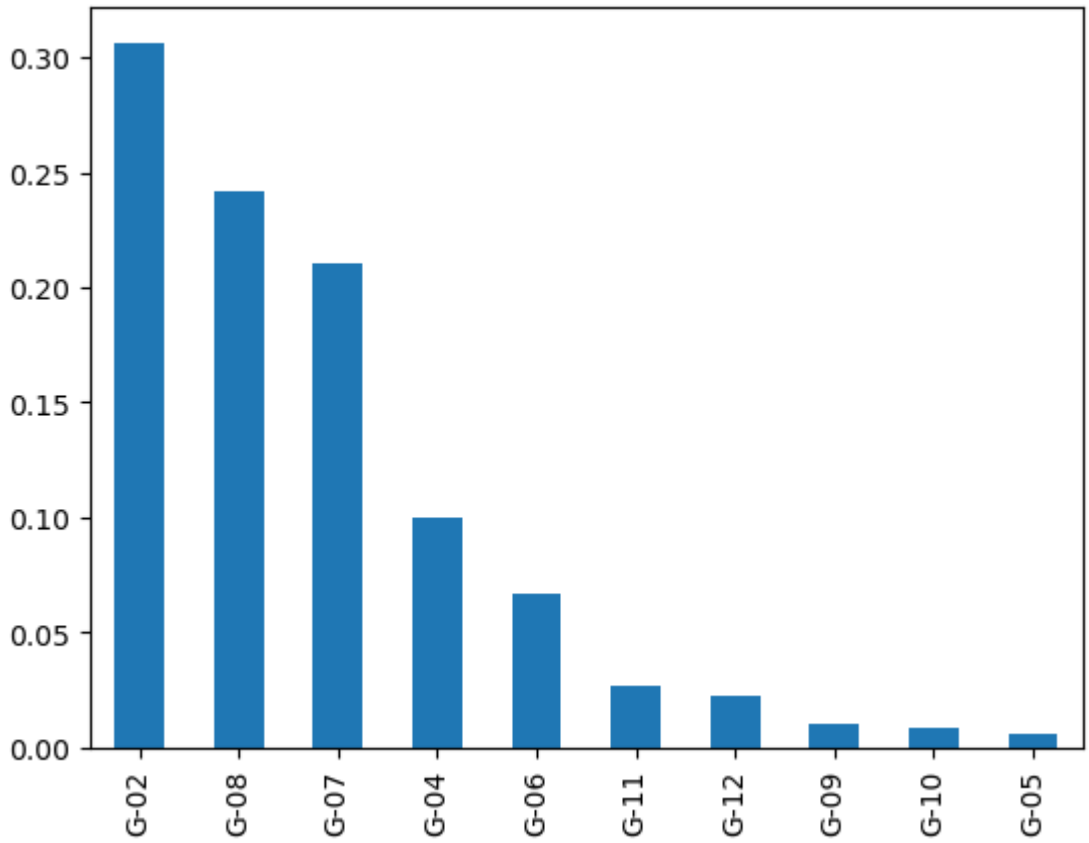
Out[35]: <Axes: >



In [36]:
```
data['GradeID'].value_counts()
print('Percentage',data.GradeID.value_counts(normalize=True))
data.GradeID.value_counts(normalize=True).plot(kind='bar')
```

```
Percentage G-02    0.306250
G-08    0.241667
G-07    0.210417
G-04    0.100000
G-06    0.066667
G-11    0.027083
G-12    0.022917
G-09    0.010417
G-10    0.008333
G-05    0.006250
Name: GradeID, dtype: float64
```

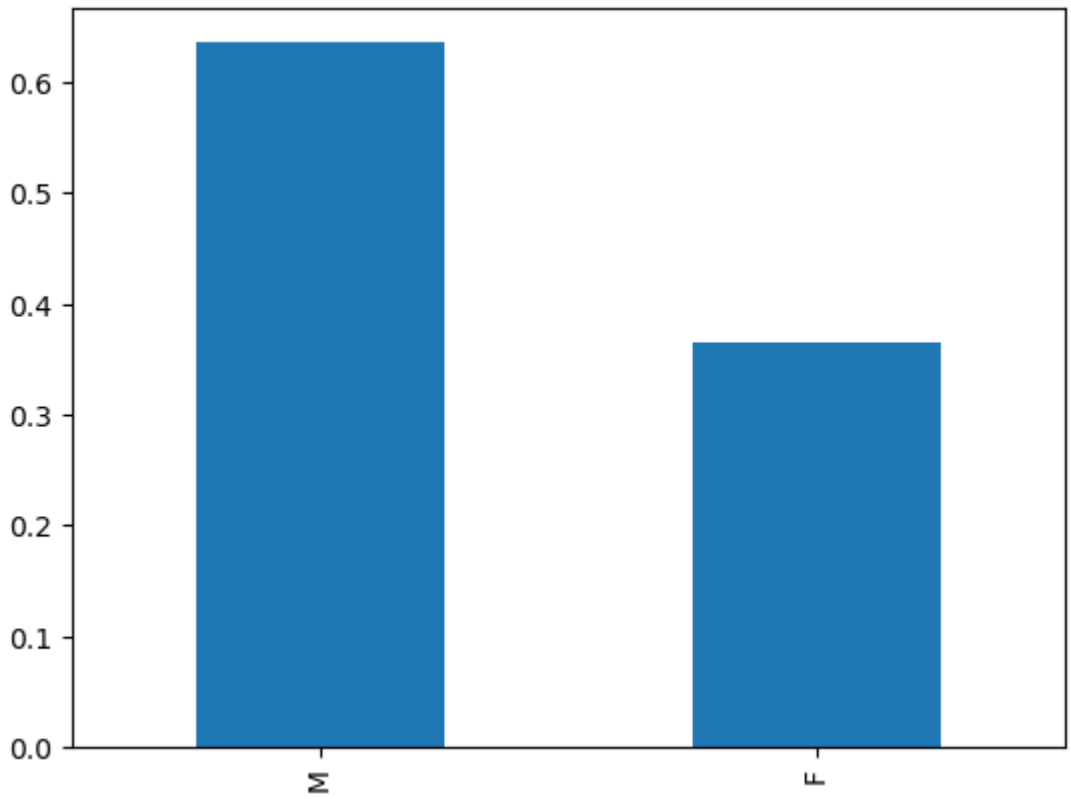
Out[36]: <Axes: >



In [37]:
```
data['gender'].value_counts()
print('Percentage',data.gender.value_counts(normalize=True))
data.gender.value_counts(normalize=True).plot(kind='bar')
```

```
Percentage M    0.635417
F    0.364583
Name: gender, dtype: float64
```

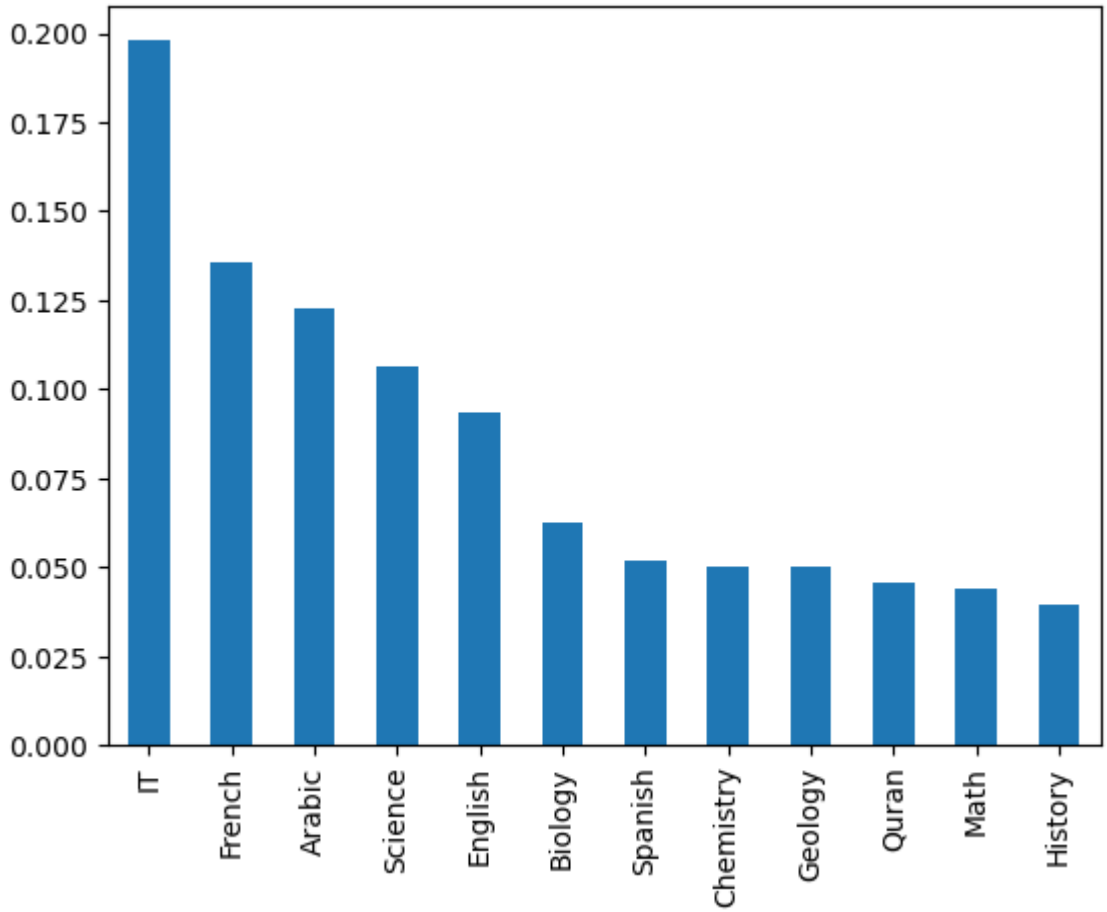
Out[37]: <Axes: >



In [38]:
```
data['Topic'].value_counts()
print('Percentage',data.Topic.value_counts(normalize=True))
data.Topic.value_counts(normalize=True).plot(kind='bar')
```

```
Percentage IT           0.197917
French       0.135417
Arabic       0.122917
Science      0.106250
English      0.093750
Biology      0.062500
Spanish      0.052083
Chemistry    0.050000
Geology      0.050000
Quran        0.045833
Math         0.043750
History      0.039583
Name: Topic, dtype: float64
```

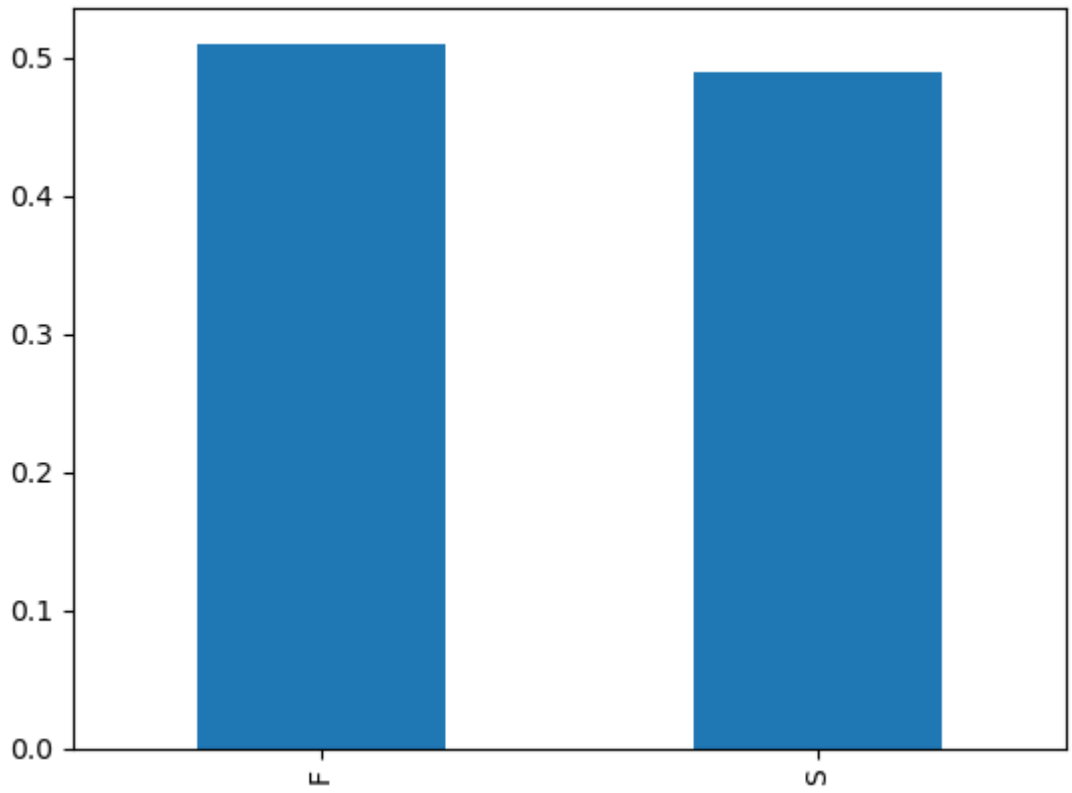
Out[38]: <Axes: >



In [39]:
```
data['Semester'].value_counts()
print('Percentage',data.Semester.value_counts(normalize=True))
data.Semester.value_counts(normalize=True).plot(kind='bar')
```

```
Percentage F    0.510417
S    0.489583
Name: Semester, dtype: float64
```

Out[39]: <Axes: >

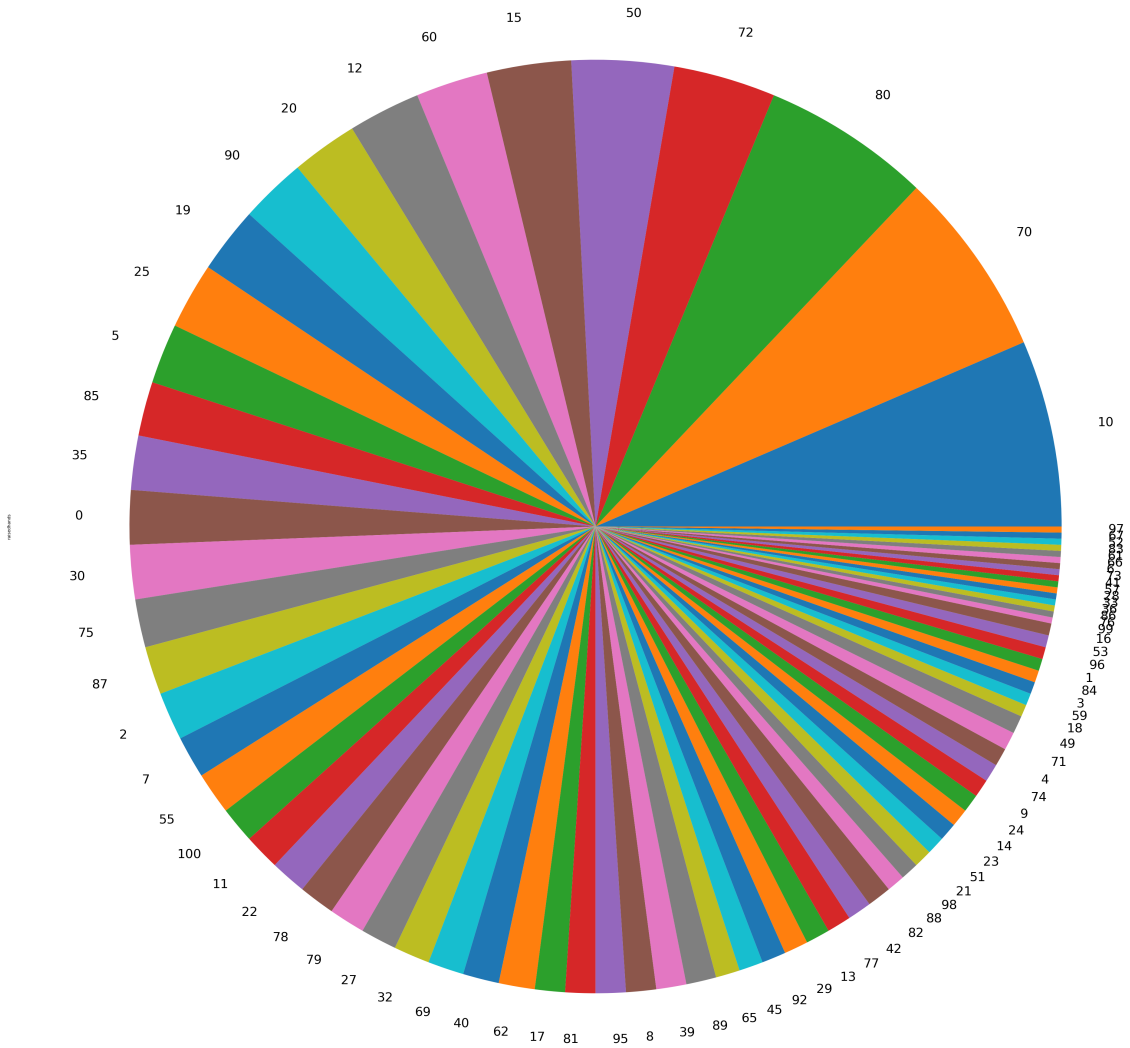


In [40]:
```
data['raisedhands'].value_counts()
print('Percentage',data.raisedhands.value_counts(normalize=True))
```

```
Percentage 10    0.064583
70    0.064583
80    0.058333
72    0.035417
50    0.035417
        ...
61    0.002083
83    0.002083
52    0.002083
67    0.002083
97    0.002083
Name: raisedhands, Length: 82, dtype: float64
```

In [41]:
```
data.raisedhands.value_counts(normalize=True).plot(kind='pie',figsize=(50,50)
```

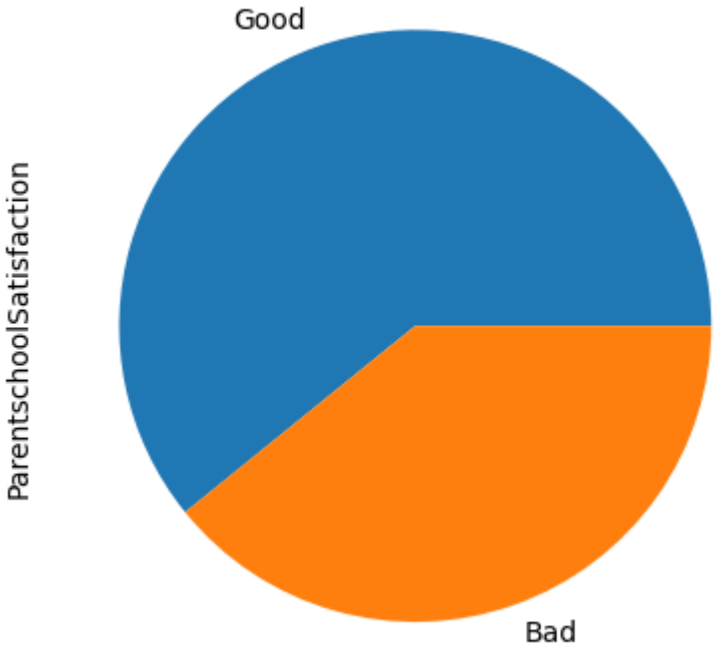
Out[41]:
```
<Axes: ylabel='raisedhands'>
```



In [42]:
```
data['ParentschoolSatisfaction'].value_counts()
print('Percentage',data.ParentschoolSatisfaction.value_counts(normalize=True))
data.ParentschoolSatisfaction.value_counts(normalize=True).plot(kind='pie')
```

```
Percentage Good    0.608333
Bad     0.391667
Name: ParentschoolSatisfaction, dtype: float64
```

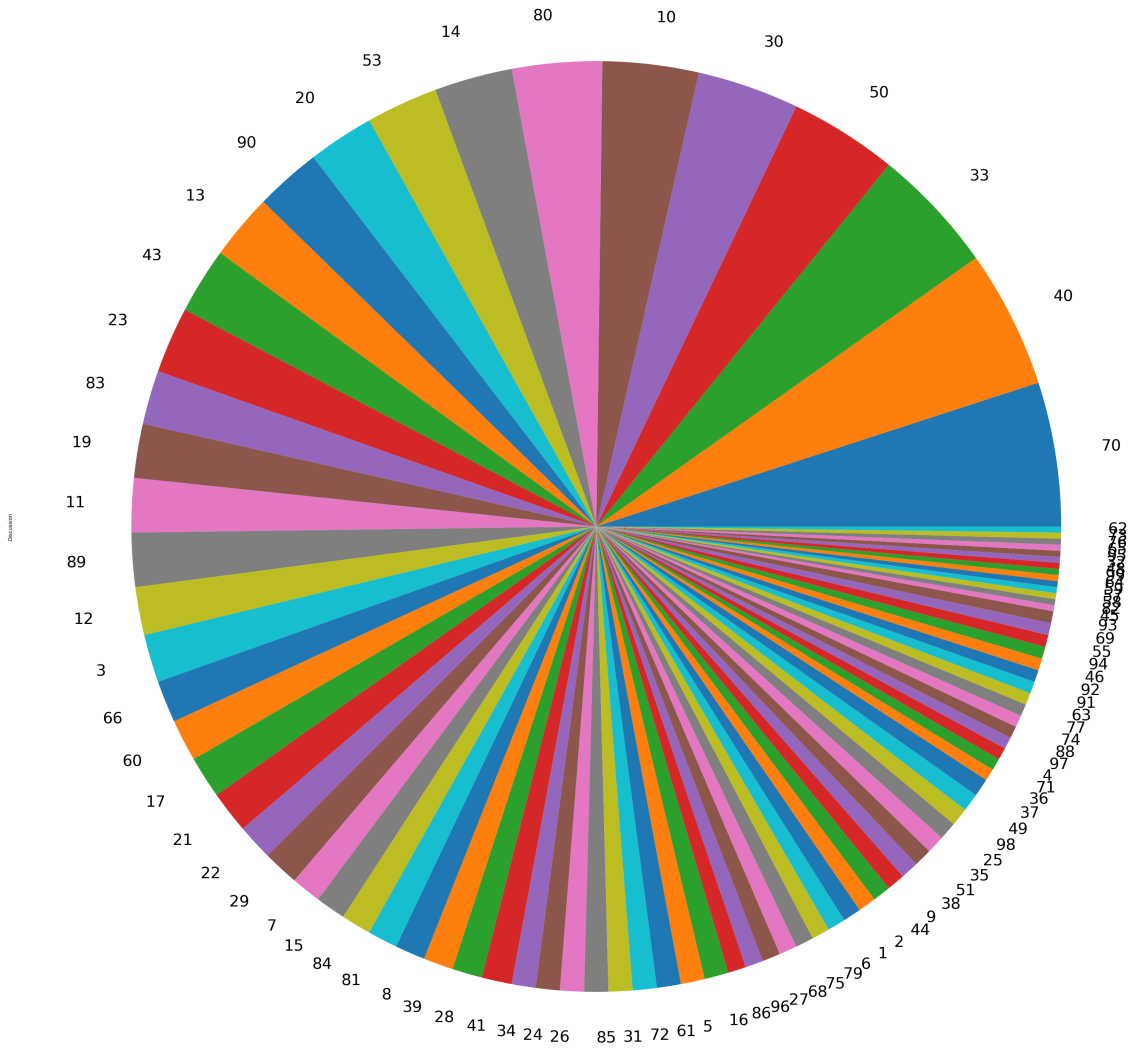
Out[42]: <Axes: ylabel='ParentschoolSatisfaction'>



In [43]:
```
data['Discussion'].value_counts()
print('Percentage',data.Discussion.value_counts(normalize=True))
data.Discussion.value_counts(normalize=True).plot(kind='pie',figsize=(40,40),
```

```
Percentage 70    0.050000
40    0.047917
33    0.043750
50    0.037500
30    0.035417
        ...
95    0.002083
65    0.002083
76    0.002083
73    0.002083
62    0.002083
Name: Discussion, Length: 90, dtype: float64
```

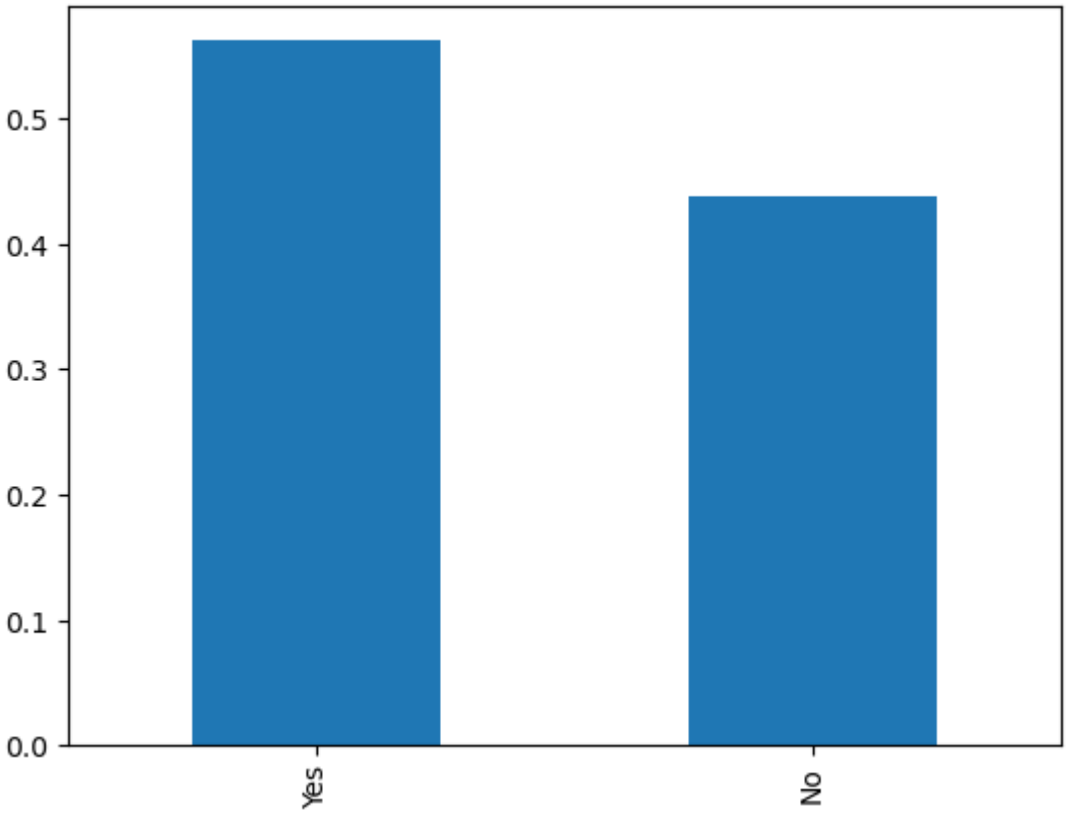
Out[43]: <Axes: ylabel='Discussion'>



In [44]:
```
data['ParentAnsweringSurvey'].value_counts()
print('Percentage',data.ParentAnsweringSurvey.value_counts(normalize=True))
data.ParentAnsweringSurvey.value_counts(normalize=True).plot(kind='bar')
```

```
Percentage Yes    0.5625
No     0.4375
Name: ParentAnsweringSurvey, dtype: float64
```

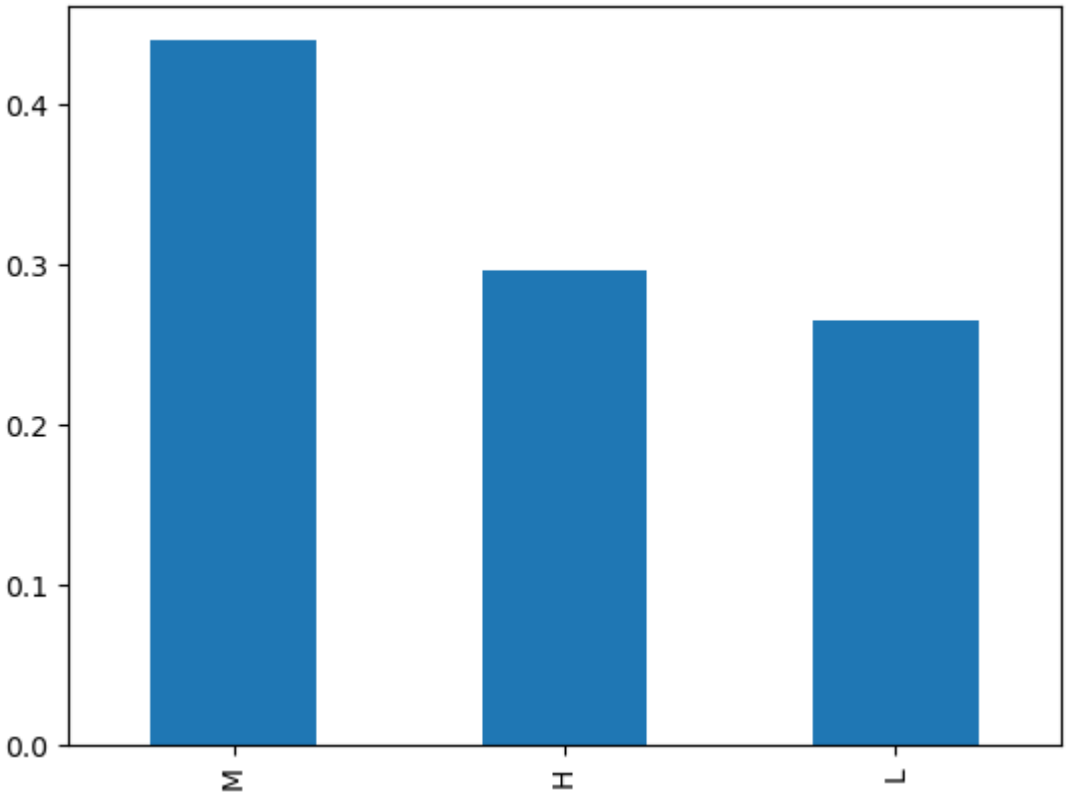
Out[44]: <Axes: >



In [45]:
```
print('Percentage',data.Class.value_counts(normalize=True))
data.Class.value_counts(normalize=True).plot(kind='bar')
```

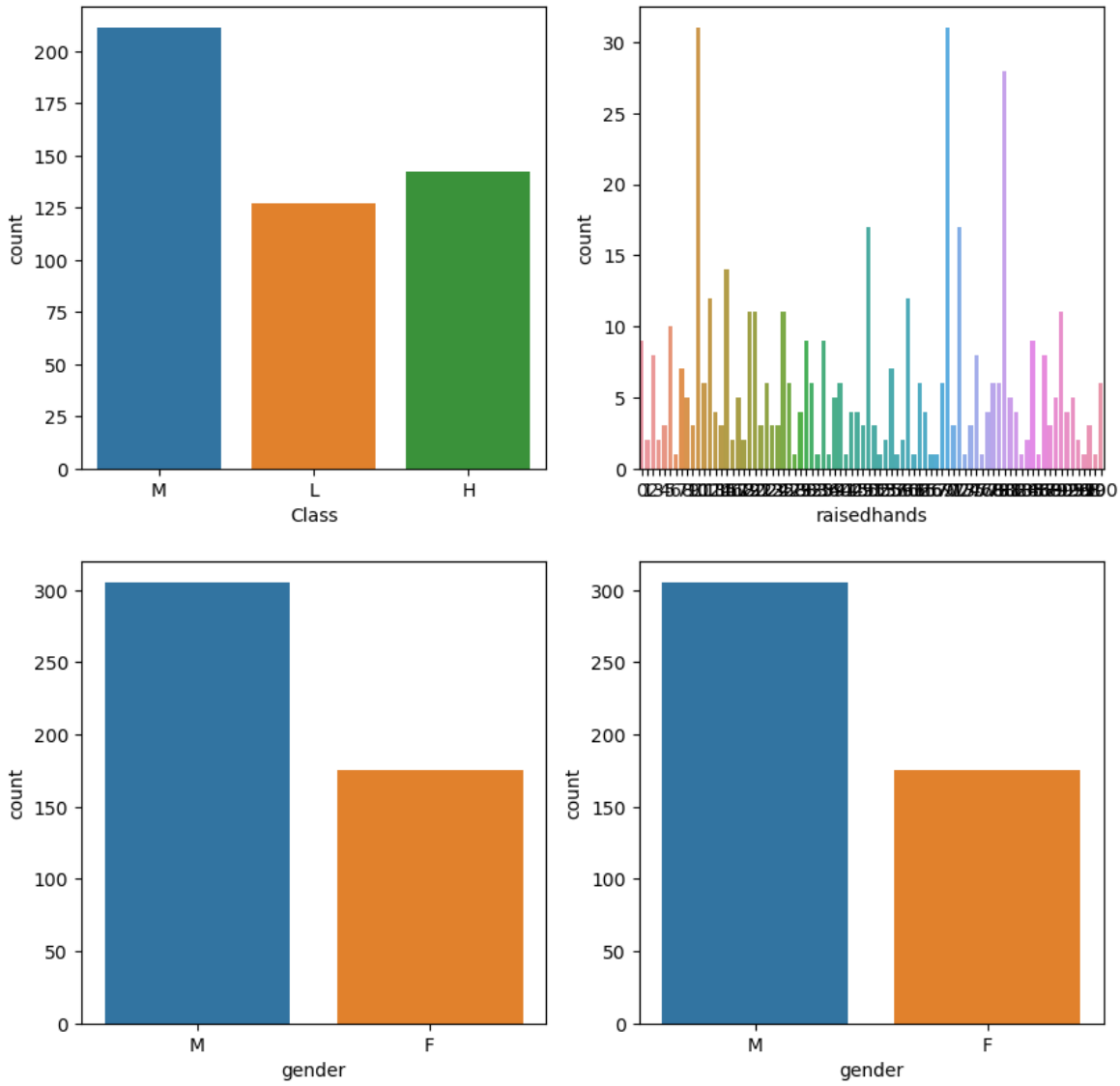
```
Percentage M    0.439583
H    0.295833
L    0.264583
Name: Class, dtype: float64
```

Out[45]: <Axes: >



In [46]:
```
fig,axarr = plt.subplots(2,2,figsize=(10,10))
sns.countplot(x='Class',data=data, ax=axarr[0,0])
sns.countplot(x='raisedhands',data=data,ax=axarr[0,1])
sns.countplot(x='gender',data=data,ax=axarr[1,0])
sns.countplot(x='gender',data=data,ax=axarr[1,1])
```

Out[46]: <Axes: xlabel='gender', ylabel='count'>



In [47]:
```
X = data.drop('Class', axis=1)
y = data['Class']
```

In [48]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rand
```

In [49]:
```
X_train
```

Out[49]:

	gender	NationalITy	PlaceofBirth	StageID	GradeID	SectionID	Topic	Semester	F
132	M	KW	KuwaIT	lowerlevel	G-02	C	IT	S	
228	M	KW	KuwaIT	HighSchool	G-11	B	Math	S	
473	M	Palestine	Palestine	MiddleSchool	G-08	A	Geology	S	
42	M	KW	KuwaIT	HighSchool	G-09	A	IT	F	
360	M	Jordan	Jordan	lowerlevel	G-02	A	Arabic	F	
...	...	...	...	...	...	...	...	...	
106	F	KW	KuwaIT	lowerlevel	G-02	B	IT	F	
270	F	Jordan	Jordan	MiddleSchool	G-06	A	English	F	
348	M	Lybia	Lybia	lowerlevel	G-02	B	French	F	
435	M	Jordan	Jordan	MiddleSchool	G-08	A	Chemistry	S	
102	F	KW	KuwaIT	lowerlevel	G-02	B	IT	F	

384 rows × 16 columns

In [50]:
```
X_test
```

Out[50]:

	gender	NationalITy	PlaceofBirth	StageID	GradeID	SectionID	Topic	Semester	Re
73	F	KW	KuwaIT	MiddleSchool	G-07	A	English	F	I
414	F	Lybia	Lybia	MiddleSchool	G-07	B	Biology	F	
394	M	Jordan	Palestine	MiddleSchool	G-07	A	Biology	F	
277	M	Palestine	Jordan	MiddleSchool	G-06	A	English	S	
399	M	Palestine	Palestine	MiddleSchool	G-07	A	Biology	S	I
...	...	...	...	...	...	...	...	...	
222	M	KW	KuwaIT	MiddleSchool	G-08	B	Spanish	S	I
237	M	KW	KuwaIT	MiddleSchool	G-07	B	Science	S	I
408	M	Jordan	Jordan	MiddleSchool	G-07	B	Biology	F	I
25	M	KW	KuwaIT	MiddleSchool	G-07	A	IT	F	I
419	M	Palestine	Jordan	MiddleSchool	G-07	B	Biology	S	I

96 rows × 16 columns

In [51]:
```
Features= data.drop('gender',axis=1)
Target=data['gender']
label=LabelEncoder()
Cat_Colums=Features.dtypes.pipe(lambda Features:Features[Features=='object'])
for col in Cat_Colums:
    Features[col]=label.fit_transform(Features[col])
    print(Features)
```

```
     NationalITy PlaceofBirth       StageID GradeID SectionID      Topic
\
0              4       KuwaIT    lowerlevel    G-04         A         IT
1              4       KuwaIT    lowerlevel    G-04         A         IT
2              4       KuwaIT    lowerlevel    G-04         A         IT
3              4       KuwaIT    lowerlevel    G-04         A         IT
4              4       KuwaIT    lowerlevel    G-04         A         IT
..           ...          ...           ...     ...       ...        ...
475            3       Jordan  MiddleSchool    G-08         A  Chemistry
476            3       Jordan  MiddleSchool    G-08         A    Geology
477            3       Jordan  MiddleSchool    G-08         A    Geology
478            3       Jordan  MiddleSchool    G-08         A    History
479            3       Jordan  MiddleSchool    G-08         A    History

    Semester Relation  raisedhands  VisITedResources  AnnouncementsView
\
0          F   Father           15                16                  2
1          F   Father           20                20                  3
2          F   Father           10                 7                  0
```

In [52]:
```
Features= data.drop('raisedhands',axis=1)
Target=data['raisedhands']
label=LabelEncoder()
Cat_Colums=Features.dtypes.pipe(lambda Features:Features[Features=='object'])
for col in Cat_Colums:
    Features[col]=label.fit_transform(Features[col])
    print(Features)
```

```
476          28                  No                  Bad
477          29                  No                  Bad
478          57                  No                  Bad
479          62                  No                  Bad

    StudentAbsenceDays Class
0              Under-7     M
1              Under-7     M
2              Above-7     L
3              Above-7     L
4              Above-7     M
..                 ...   ...
475            Above-7     L
476            Under-7     M
477            Under-7     M
478            Above-7     L
479            Above-7     L

[480 rows x 16 columns]
     gender  NationalITy  PlaceofBirth  StageID  GradeID  SectionID  Topi
```

In [53]:
```
X = data.drop('Class', axis=1)
y = data['Class']
```

In [54]:
```
X = pd.get_dummies(X)
```

In [55]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, ran
```

In [56]:
```
x_train,x_test,y_train,y_test=train_test_split(Features,Target,test_size=0.2,
print(x_train)
print(x_test)
print(y_train)
print(y_test)
```

```
     gender  NationalITy  PlaceofBirth  StageID  GradeID  SectionID  Topi
c  \
299       1            3             3        2        1          0     1
0
348       1            5             5        2        0          1
4
162       0            3             3        2        0          1
4
467       0            3             3        1        5          0
5
306       1            3             3        2        1          1     1
0
..      ...          ...           ...      ...      ...        ...
...
86        1            8             8        2        0          1
7
151       1            8            11        0        8          0     1
0
13        1           12            12        1        5          0
```

In [57]:
```
Logit_Model=LogisticRegression()
Logit_Model.fit(x_train,y_train)
```

```
D:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:458: Converg
enceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scik
it-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regre
ssion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-re
gression)
  n_iter_i = _check_optimize_result(
```

Out[57]:
```
LogisticRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [58]:
```
from sklearn.metrics import confusion_matrix,classification_report,accuracy_s
Prediction=Logit_Model.predict(x_test)
score = accuracy_score(y_test,Prediction)
Report=classification_report(y_test,Prediction)
```

```
D:\Anaconda\lib\site-packages\sklearn\metrics\_classification.py:1344: Undef
inedMetricWarning: Precision and F-score are ill-defined and being set to 0.
0 in labels with no predicted samples. Use `zero_division` parameter to cont
rol this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
D:\Anaconda\lib\site-packages\sklearn\metrics\_classification.py:1344: Undef
inedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 i
n labels with no true samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
D:\Anaconda\lib\site-packages\sklearn\metrics\_classification.py:1344: Undef
inedMetricWarning: Precision and F-score are ill-defined and being set to 0.
0 in labels with no predicted samples. Use `zero_division` parameter to cont
rol this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
D:\Anaconda\lib\site-packages\sklearn\metrics\_classification.py:1344: Undef
inedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 i
n labels with no true samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
D:\Anaconda\lib\site-packages\sklearn\metrics\_classification.py:1344: Undef
inedMetricWarning: Precision and F-score are ill-defined and being set to 0.
0 in labels with no predicted samples. Use `zero_division` parameter to cont
rol this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
D:\Anaconda\lib\site-packages\sklearn\metrics\_classification.py:1344: Undef
inedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 i
n labels with no true samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [59]:
```
print(Prediction)
```

```
[  5  70  29   0  85  17  80  85  10  10  10   0  70   0  70  10  79  19
  70  72  50  79   7  49  75  12  70  90  90  10  13  70   7  70   2  10
  10   7  70  20  70  70  81  60  70   0  81  17  90   2 100  62  51  50
  70  10  85  10   0  85  10  70  85  50  70  85  10  10  81  70  70  72
  81  81  50  70  60  90  62  40  10  79  72  70  10  10  50  62  22  81
  10  10  12  70  87  79]
```

In [60]:
```
print('score :',score)
```

```
score : 0.041666666666666664
```

In [61]:
```
print('Report',Report)
```

Report	precision	recall	f1-score	support
0	0.00	0.00	0.00	0
1	0.00	0.00	0.00	1
2	0.00	0.00	0.00	2
4	0.00	0.00	0.00	1
5	0.00	0.00	0.00	3
7	0.00	0.00	0.00	2
8	0.00	0.00	0.00	1
9	0.00	0.00	0.00	1
10	0.06	0.33	0.10	3
11	0.00	0.00	0.00	1
12	0.00	0.00	0.00	1
13	0.00	0.00	0.00	0
14	0.00	0.00	0.00	1
17	0.00	0.00	0.00	1
19	0.00	0.00	0.00	2
20	0.00	0.00	0.00	2
21	0.00	0.00	0.00	2
22	0.00	0.00	0.00	1
25	0.00	0.00	0.00	3
27	0.00	0.00	0.00	2
29	0.00	0.00	0.00	0
30	0.00	0.00	0.00	1
32	0.00	0.00	0.00	2
35	0.00	0.00	0.00	1
39	0.00	0.00	0.00	1
40	0.00	0.00	0.00	2
42	0.00	0.00	0.00	1
49	0.00	0.00	0.00	0
50	0.00	0.00	0.00	2
51	0.00	0.00	0.00	1
52	0.00	0.00	0.00	1
53	0.00	0.00	0.00	1
55	0.00	0.00	0.00	3
59	0.00	0.00	0.00	1
60	0.00	0.00	0.00	2
62	0.00	0.00	0.00	1
69	0.00	0.00	0.00	1
70	0.11	0.29	0.15	7
72	0.00	0.00	0.00	4
74	0.00	0.00	0.00	1
75	0.00	0.00	0.00	2
78	0.00	0.00	0.00	1
79	0.00	0.00	0.00	1
80	0.00	0.00	0.00	11
81	0.00	0.00	0.00	0
82	0.00	0.00	0.00	1
83	0.00	0.00	0.00	1
84	0.00	0.00	0.00	1
85	0.17	0.50	0.25	2
87	0.00	0.00	0.00	2
89	0.00	0.00	0.00	3
90	0.00	0.00	0.00	3
95	0.00	0.00	0.00	1
97	0.00	0.00	0.00	1
98	0.00	0.00	0.00	2

```
           100       0.00      0.00      0.00         1

    accuracy                           0.04        96
   macro avg       0.01      0.02      0.01        96
weighted avg       0.01      0.04      0.02        96
```

In [62]:
```
Logit_Model=LinearRegression()
Logit_Model.fit(x_train,y_train)
```

Out[62]:
```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [66]:
```
print(Prediction)
```

```
[ 5.17530222 53.64098014  7.85957764 29.25402835 47.59378428 20.92043995
 60.00715118 67.3512335  43.26275075 35.00203798 42.53261436 10.8163463
 64.30640574  8.96417171 56.03720415 18.48832232 73.40423283 43.82931317
 58.43478028 64.99689338 50.97417771 60.0304731  15.58465278 29.47371029
 68.76590773 14.80591505 65.31678083 69.45775607 80.42512331 14.15372614
 55.39291505 59.6109234  15.2982079  73.45351154 19.23219817 36.74136495
  9.90354153 17.57056997 94.19198509 41.3053664  73.89560689 68.57426925
 78.63082439 54.22289873 75.92737301  5.92722294 62.57054531 12.17479331
 57.9205366  22.35200019 83.32475563 87.58333719 52.95875818 49.99276128
 53.15134667 26.08918572 73.36287495 16.98316318  4.87365367 55.0968661
 24.53435864 76.59155228 71.5846938  40.94315509 60.79290974 60.24223792
 30.86494723 23.58823861 88.00651597 56.59807648 74.25002912 81.06086105
 80.06513772 75.51670287 66.43483076 86.95591421 61.09828378 79.79288998
 65.8619451  42.21137943 16.80307137 82.32206658 75.8283572  55.7762945
  6.82442138 31.55543132 47.89009733 86.53567147 32.49824061 80.74361797
  9.12899369 17.05008748 42.31530428 53.73960508 71.84203415 70.50634792]
```

In [67]:
```
print('score :',score)
```

```
score : 0.041666666666666664
```

In [68]:
```
print('Report',Report)
```

Report	precision	recall	f1-score	support
0	0.00	0.00	0.00	0
1	0.00	0.00	0.00	1
2	0.00	0.00	0.00	2
4	0.00	0.00	0.00	1
5	0.00	0.00	0.00	3
7	0.00	0.00	0.00	2
8	0.00	0.00	0.00	1
9	0.00	0.00	0.00	1
10	0.06	0.33	0.10	3
11	0.00	0.00	0.00	1
12	0.00	0.00	0.00	1
13	0.00	0.00	0.00	0
14	0.00	0.00	0.00	1
17	0.00	0.00	0.00	1
19	0.00	0.00	0.00	2
20	0.00	0.00	0.00	2
21	0.00	0.00	0.00	2
22	0.00	0.00	0.00	1
25	0.00	0.00	0.00	3
27	0.00	0.00	0.00	2
29	0.00	0.00	0.00	0
30	0.00	0.00	0.00	1
32	0.00	0.00	0.00	2
35	0.00	0.00	0.00	1
39	0.00	0.00	0.00	1
40	0.00	0.00	0.00	2
42	0.00	0.00	0.00	1
49	0.00	0.00	0.00	0
50	0.00	0.00	0.00	2
51	0.00	0.00	0.00	1
52	0.00	0.00	0.00	1
53	0.00	0.00	0.00	1
55	0.00	0.00	0.00	3
59	0.00	0.00	0.00	1
60	0.00	0.00	0.00	2
62	0.00	0.00	0.00	1
69	0.00	0.00	0.00	1
70	0.11	0.29	0.15	7
72	0.00	0.00	0.00	4
74	0.00	0.00	0.00	1
75	0.00	0.00	0.00	2
78	0.00	0.00	0.00	1
79	0.00	0.00	0.00	1
80	0.00	0.00	0.00	11
81	0.00	0.00	0.00	0
82	0.00	0.00	0.00	1
83	0.00	0.00	0.00	1
84	0.00	0.00	0.00	1
85	0.17	0.50	0.25	2
87	0.00	0.00	0.00	2
89	0.00	0.00	0.00	3
90	0.00	0.00	0.00	3
95	0.00	0.00	0.00	1
97	0.00	0.00	0.00	1
98	0.00	0.00	0.00	2

```
                  100       0.00      0.00      0.00         1

             accuracy                           0.04        96
            macro avg       0.01      0.02      0.01        96
         weighted avg       0.01      0.04      0.02        96
```

In [ ]:

In [ ]: