In [2]:
```python
#IMPORTING LIBRARIES
import pandas as pd
import numpy as np
import seaborn as sns
```

In [3]:
```python
#IMPORTING DATASET
df= pd.read_csv("pd_speech_features.csv")
df
```

Out[3]:

	id	gender	PPE	DFA	RPDE	numPulses	numPeriodsPulses	meanPeriodPulses	stdDevPeriodPulses	locPctJitter	...	tqwt_
0	0	1	0.85247	0.71826	0.57227	240	239	0.008064	0.000087	0.00218	...	
1	0	1	0.76686	0.69481	0.53966	234	233	0.008258	0.000073	0.00195	...	
2	0	1	0.85083	0.67604	0.58982	232	231	0.008340	0.000060	0.00176	...	
3	1	0	0.41121	0.79672	0.59257	178	177	0.010858	0.000183	0.00419	...	
4	1	0	0.32790	0.79782	0.53028	236	235	0.008162	0.002669	0.00535	...	
...	...	...	...	...	...	...	...	...	...	...	...	
751	250	0	0.80903	0.56355	0.28385	417	416	0.004627	0.000052	0.00064	...	
752	250	0	0.16084	0.56499	0.59194	415	413	0.004550	0.000220	0.00143	...	
753	251	0	0.88389	0.72335	0.46815	381	380	0.005069	0.000103	0.00076	...	
754	251	0	0.83782	0.74890	0.49823	340	339	0.005679	0.000055	0.00092	...	
755	251	0	0.81304	0.76471	0.46374	340	339	0.005676	0.000037	0.00078	...	

756 rows × 755 columns

In [4]:
```python
df.head()
```

Out[4]:

	id	gender	PPE	DFA	RPDE	numPulses	numPeriodsPulses	meanPeriodPulses	stdDevPeriodPulses	locPctJitter	...	tqwt_kurt
0	0	1	0.85247	0.71826	0.57227	240	239	0.008064	0.000087	0.00218	...	
1	0	1	0.76686	0.69481	0.53966	234	233	0.008258	0.000073	0.00195	...	
2	0	1	0.85083	0.67604	0.58982	232	231	0.008340	0.000060	0.00176	...	
3	1	0	0.41121	0.79672	0.59257	178	177	0.010858	0.000183	0.00419	...	
4	1	0	0.32790	0.79782	0.53028	236	235	0.008162	0.002669	0.00535	...	

5 rows × 755 columns

In [5]:
```python
df.tail()
```

Out[5]:

	id	gender	PPE	DFA	RPDE	numPulses	numPeriodsPulses	meanPeriodPulses	stdDevPeriodPulses	locPctJitter	...	tqwt_
751	250	0	0.80903	0.56355	0.28385	417	416	0.004627	0.000052	0.00064	...	
752	250	0	0.16084	0.56499	0.59194	415	413	0.004550	0.000220	0.00143	...	
753	251	0	0.88389	0.72335	0.46815	381	380	0.005069	0.000103	0.00076	...	
754	251	0	0.83782	0.74890	0.49823	340	339	0.005679	0.000055	0.00092	...	
755	251	0	0.81304	0.76471	0.46374	340	339	0.005676	0.000037	0.00078	...	

5 rows × 755 columns

In [6]:
```python
df.size
```

Out[6]:
```
570780
```

In [7]:
```python
df.shape
```

Out[7]:
```
(756, 755)
```

In [8]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 756 entries, 0 to 755
Columns: 755 entries, id to class
dtypes: float64(749), int64(6)
memory usage: 4.4 MB
```

In [9]:
```
df.describe()
```

Out[9]:

	id	gender	PPE	DFA	RPDE	numPulses	numPeriodsPulses	meanPeriodPulses	stdDevPeriodPulses
count	756.000000	756.000000	756.000000	756.000000	756.000000	756.000000	756.000000	756.000000	756.00000
mean	125.500000	0.515873	0.746284	0.700414	0.489058	323.972222	322.678571	0.006360	0.00038
std	72.793721	0.500079	0.169294	0.069718	0.137442	99.219059	99.402499	0.001826	0.00072
min	0.000000	0.000000	0.041551	0.543500	0.154300	2.000000	1.000000	0.002107	0.000011
25%	62.750000	0.000000	0.762833	0.647053	0.386537	251.000000	250.000000	0.005003	0.00004
50%	125.500000	1.000000	0.809655	0.700525	0.484355	317.000000	316.000000	0.006048	0.000077
75%	188.250000	1.000000	0.834315	0.754985	0.586515	384.250000	383.250000	0.007528	0.000171
max	251.000000	1.000000	0.907660	0.852640	0.871230	907.000000	905.000000	0.012966	0.00348

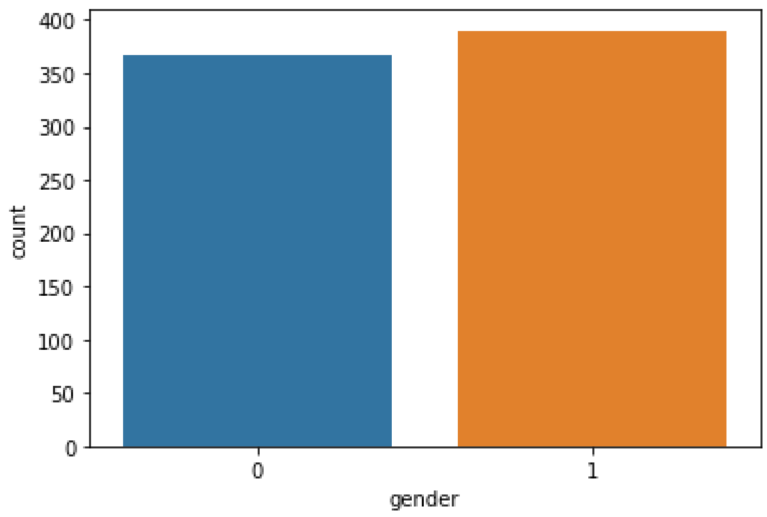
8 rows × 755 columns

In [10]:
```
df.isnull().sum()
```

Out[10]:
```
id                           0
gender                       0
PPE                          0
DFA                          0
RPDE                         0
                            ..
tqwt_kurtosisValue_dec_33    0
tqwt_kurtosisValue_dec_34    0
tqwt_kurtosisValue_dec_35    0
tqwt_kurtosisValue_dec_36    0
class                        0
Length: 755, dtype: int64
```

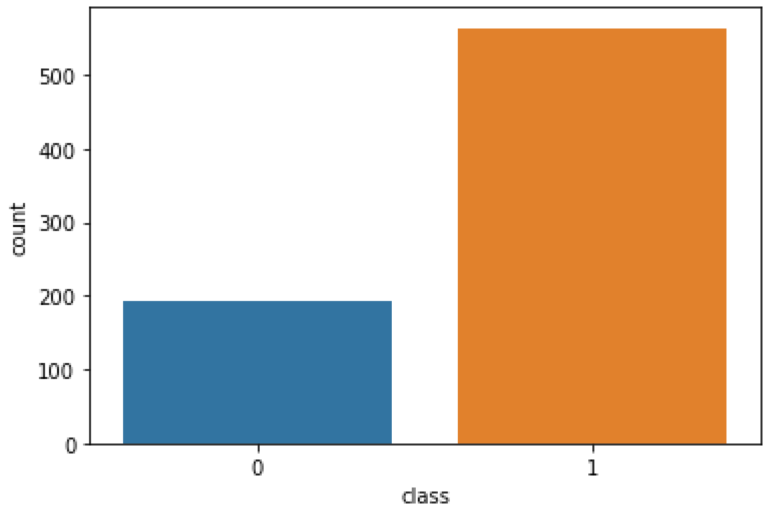
In [11]:
```
#EDA
sns.countplot(df['gender'])
```

Out[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x27974dbc3a0>
```



In [12]:
```
sns.countplot(df['class'])
```

Out[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x27974dbcf10>
```



In [13]:

```
#Data pre-processing
x = df.drop(['class'], 1)
x
```

Out[13]:

	id	gender	PPE	DFA	RPDE	numPulses	numPeriodsPulses	meanPeriodPulses	stdDevPeriodPulses	locPctJitter	...	tqwt_
0	0	1	0.85247	0.71826	0.57227	240	239	0.008064	0.000087	0.00218	...	
1	0	1	0.76686	0.69481	0.53966	234	233	0.008258	0.000073	0.00195	...	
2	0	1	0.85083	0.67604	0.58982	232	231	0.008340	0.000060	0.00176	...	
3	1	0	0.41121	0.79672	0.59257	178	177	0.010858	0.000183	0.00419	...	
4	1	0	0.32790	0.79782	0.53028	236	235	0.008162	0.002669	0.00535	...	
...	...	...	...	...	...	...	...	...	...	...	...	
751	250	0	0.80903	0.56355	0.28385	417	416	0.004627	0.000052	0.00064	...	
752	250	0	0.16084	0.56499	0.59194	415	413	0.004550	0.000220	0.00143	...	
753	251	0	0.88389	0.72335	0.46815	381	380	0.005069	0.000103	0.00076	...	
754	251	0	0.83782	0.74890	0.49823	340	339	0.005679	0.000055	0.00092	...	
755	251	0	0.81304	0.76471	0.46374	340	339	0.005676	0.000037	0.00078	...	

756 rows × 754 columns

In [14]:

```
y = df['class']
y
```

Out[14]:

```
0      1
1      1
2      1
3      1
4      1
      ..
751    0
752    0
753    0
754    0
755    0
Name: class, Length: 756, dtype: int64
```

In [15]:

```
#TRAIN TEST SPLIT
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 42)
```

In [16]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
```

In [17]:

```
print(x_train.shape)
print(x_test.shape)
```

```
(604, 754)
(152, 754)
```

In [18]:

```
#LOGISTIC REGRESSION
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state=42)
classifier.fit(x_train,y_train)
```

```
C:\Users\asus\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:762: ConvergenceWarning: 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://scikit-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-regression (https://scikit-learn.org/stable/modules/lin
ear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[18]:

```
LogisticRegression(random_state=42)
```

In [19]:

```
y_pred = classifier.predict(x_test)
```

In [20]:

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)*100
```

Out[20]:

```
73.02631578947368
```

```
In [ ]:
```