In [2]:
```
import pandas as pd
```

In [3]:
```
df = pd.read_csv('./DASS_data_21.02.19/data.csv', sep='\t')
df.head()
```

Out[3]:

	Q1A	Q1I	Q1E	Q2A	Q2I	Q2E	Q3A	Q3I	Q3E	Q4A	...	screensize	uniquenetworklocatio
0	4	28	3890	4	25	2122	2	16	1944	4	...	1	
1	4	2	8118	1	36	2890	2	35	4777	3	...	2	
2	3	7	5784	1	33	4373	4	41	3242	1	...	2	
3	2	23	5081	3	11	6837	2	37	5521	1	...	2	
4	2	36	3215	2	13	7731	3	5	4156	4	...	2	

5 rows × 172 columns

In [5]:
```
questions = [f'Q{i}A' for i in range(1, 43)]
das_df = df[questions]
das_df.head()
```

Out[5]:

	Q1A	Q2A	Q3A	Q4A	Q5A	Q6A	Q7A	Q8A	Q9A	Q10A	...	Q33A	Q34A	Q35A	Q36A	Q
0	4	4	2	4	4	4	4	4	2	1	...	2	3	4	4	
1	4	1	2	3	4	4	3	4	3	2	...	3	2	2	3	
2	3	1	4	1	4	3	1	3	2	4	...	1	4	3	4	
3	2	3	2	1	3	3	4	2	3	3	...	2	4	1	1	
4	2	2	3	4	4	2	4	4	4	3	...	4	4	3	4	

5 rows × 42 columns

***Anxiety*:**

In [9]:
```
anxiety_questions = ['Q2A','Q4A','Q7A','Q9A','Q15A','Q19A','Q20A','Q23A','Q25A
anxiety_df = das_df[anxiety_questions]
anxiety_df.head()
```

Out[9]:

	Q2A	Q4A	Q7A	Q9A	Q15A	Q19A	Q20A	Q23A	Q25A	Q28A	Q30A	Q36A	Q40A	Q41A
0	4	4	4	2	4	3	3	4	4	3	2	4	3	4
1	1	3	3	3	3	1	1	1	2	4	3	3	1	2
2	1	1	1	2	4	2	1	2	2	1	2	4	2	1
3	3	1	4	3	2	1	2	1	1	1	3	1	4	4
4	2	4	4	4	4	4	4	4	4	4	4	4	4	4

In [46]:
```
min(anxiety_df.drop('level',axis=1).sum(axis=1))
```

Out[46]: 14

In [47]:
```
max(anxiety_df.drop('level',axis=1).sum(axis=1))
```

Out[47]: 56

Intuition behind feature engineering the target variable:

since the questionaire had responses corresponding to depression symptoms in scale of 1 to 4 or 1 to 10 so lower the scale point response lower chances of depression and higher the scale point response higher chances of depression and when a patient's responses are observed over different aspects/features case 1: when almost all responses are at lower end 1 or 2 then the average/overall sum would also be low case 2: when almost all responses are at higher end 4 or 5 then the average/overall sum would also be high case 3: when almost all responses are at middle end 3 then the average/overall sum would also be in middle

It seems that sums of scale points for all patients is normaly distributed, so Now let's split it to five categories

1. Normal
2. Mild
3. Moderate
4. Severe
5. Extremely Severe

In [50]:
```
def map_anxiety_level(score):
    if score >= 49: return 'extremely severe'
    elif 40 <= score <= 48 : return 'severe'
    elif 31 <= score <= 39 : return 'moderate'
    elif 22 <= score <= 30: return 'mild'
    else: return 'normal'
anxiety_df['level'] = anxiety_df.sum(axis=1).apply(map_anxiety_level)
anxiety_df.head()
```

```
C:\Users\aniru\AppData\Local\Temp\ipykernel_26340\2003375728.py:7: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  anxiety_df['level'] = anxiety_df.sum(axis=1).apply(map_anxiety_level)
```

Out[50]:

	Q2A	Q4A	Q7A	Q9A	Q15A	Q19A	Q20A	Q23A	Q25A	Q28A	Q30A	Q36A	Q40A	Q41A
0	4	4	4	2	4	3	3	4	4	3	2	4	3	4
1	1	3	3	3	3	1	1	1	2	4	3	3	1	2
2	1	1	1	2	4	2	1	2	2	1	2	4	2	1
3	3	1	4	3	2	1	2	1	1	1	3	1	4	4
4	2	4	4	4	4	4	4	4	4	4	4	4	4	4

In [99]:
```
anxiety_features = anxiety_df.drop('level',axis=1)
anxiety_label = anxiety_df['level']
```

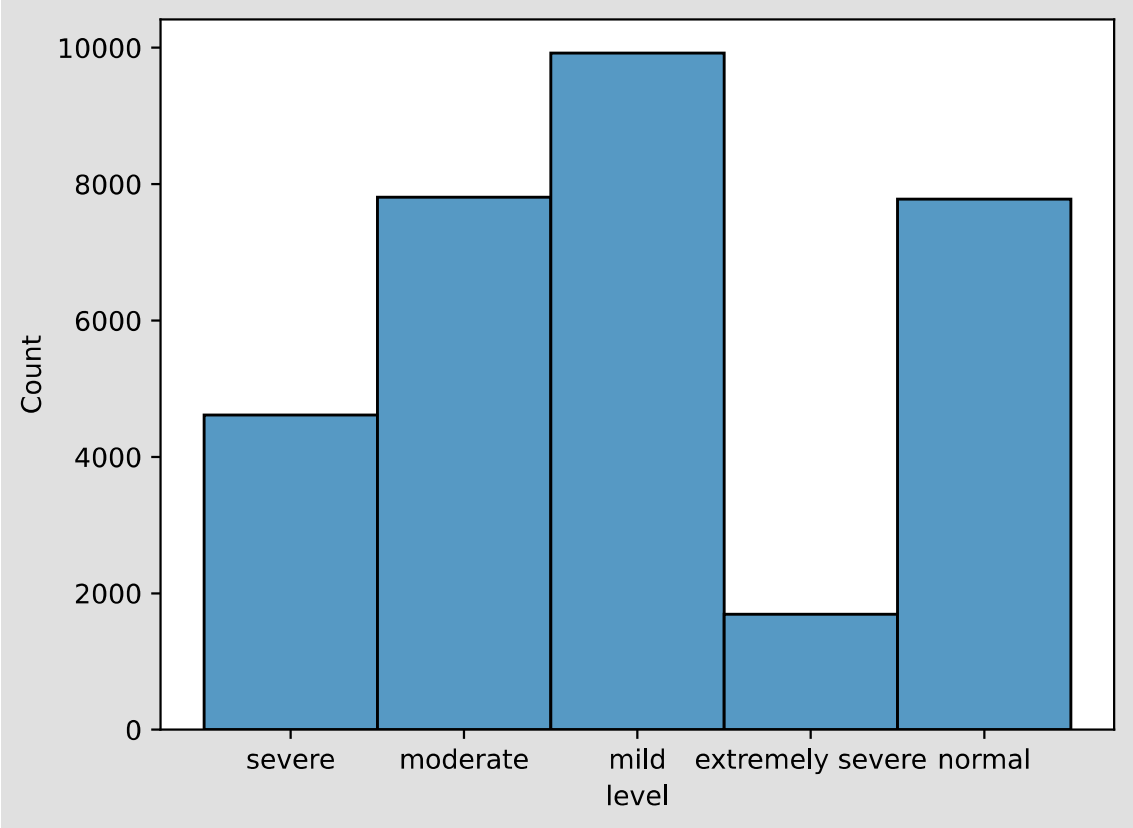
In [113]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(anxiety_features, anxiety_
                                                    stratify=anxiety_label,
                                                    test_size=0.2,random_state=
```

In [117]:
```
X_train['level'] = y_train
```

In [106]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

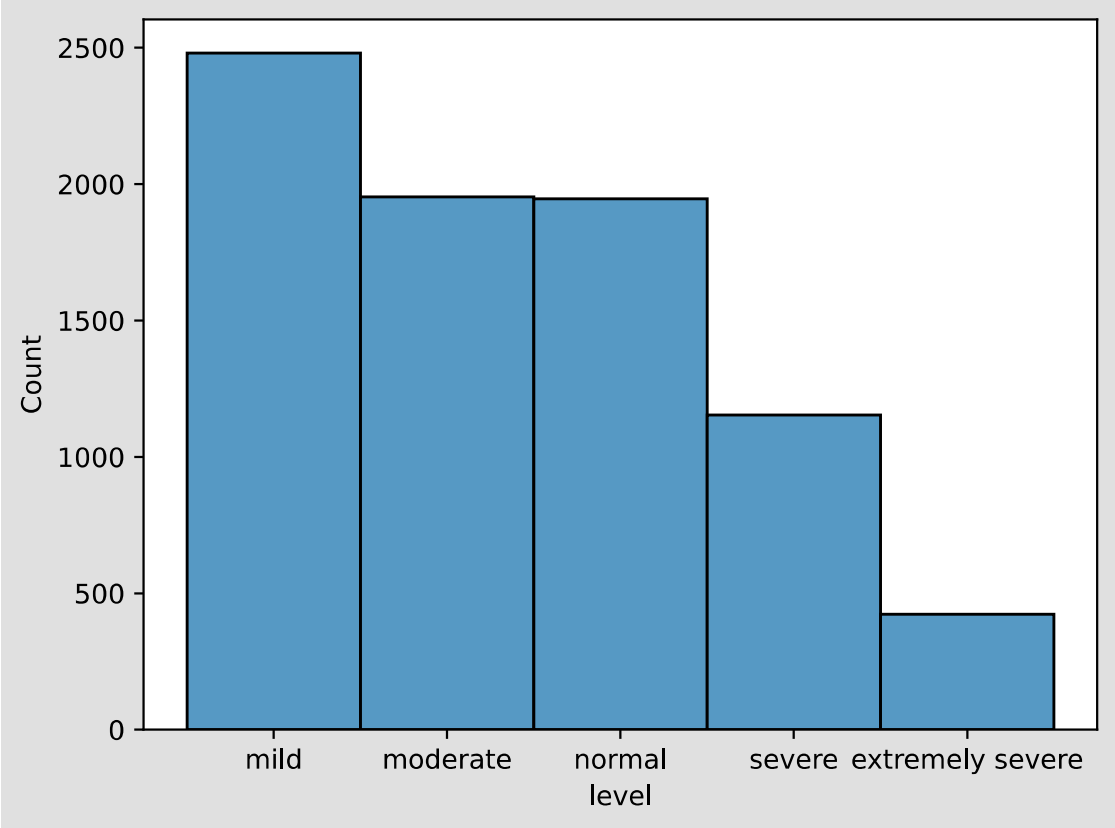
In [114]:
```
sns.histplot(x=y_train)
```

Out[114]:
```
<Axes: xlabel='level', ylabel='Count'>
```



In [115]:
```
sns.histplot(x=y_test)
```

Out[115]:
```
<Axes: xlabel='level', ylabel='Count'>
```



In [8]:
```
"A','".join('Q2 Q4 Q7 Q9 Q15 Q19 Q20 Q23 Q25 Q28 Q30 Q36 Q40 Q41'.split())
```

Out[8]:
```
"Q2A','Q4A','Q7A','Q9A','Q15A','Q19A','Q20A','Q23A','Q25A','Q28A','Q30A','Q36
A','Q40A','Q41"
```

Bayesian Network for Anxiety:

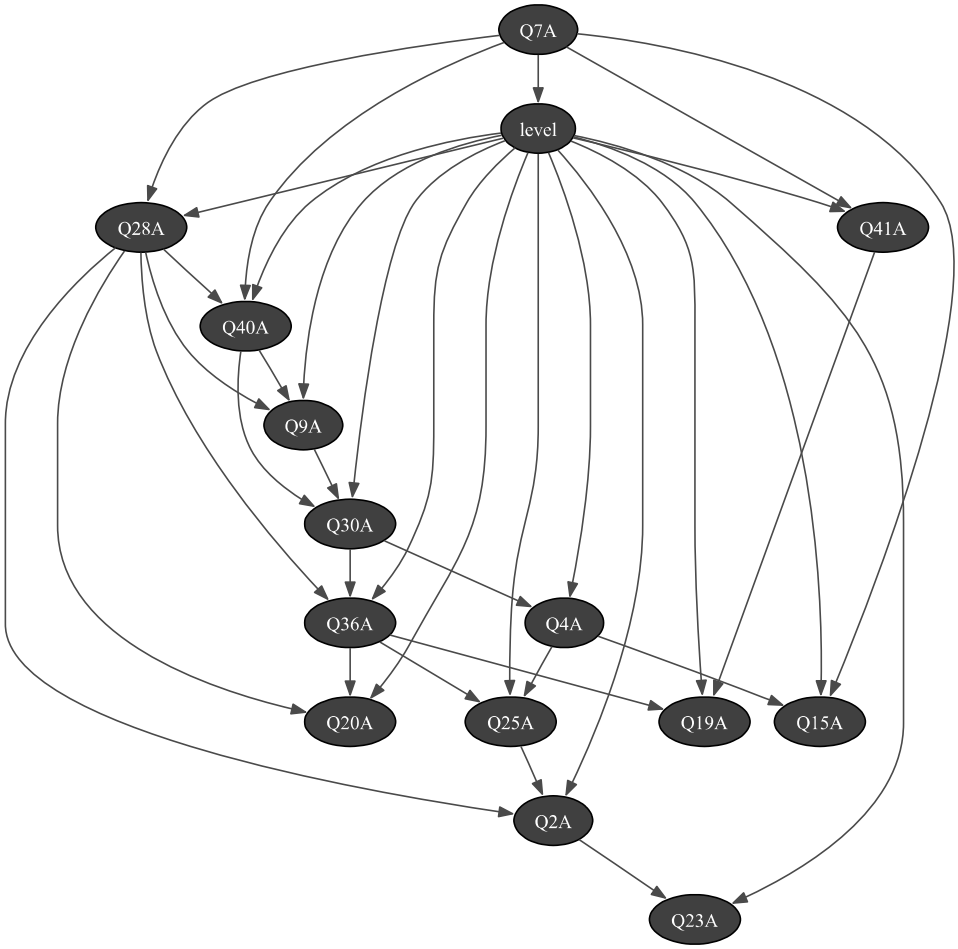
In [169]:
```
from pgmpy.estimators import HillClimbSearch
from pgmpy.estimators import BicScore
from pgmpy.models import BayesianNetwork

# Search for the best structure
hc = HillClimbSearch(X_train)
best_model = hc.estimate()

# Instantiate a BayesianModel object with the best structure
bayesian_network = BayesianNetwork(best_model.edges())

print(bayesian_network.edges())
```

0% 36/1000000 [00:04<25:42:02, 10.81it/s]

```
[('Q2A', 'Q23A'), ('Q4A', 'Q25A'), ('Q4A', 'Q15A'), ('Q25A', 'Q2A'), ('Q7A',
'level'), ('Q7A', 'Q41A'), ('Q7A', 'Q15A'), ('Q7A', 'Q28A'), ('Q7A', 'Q40A'),
('level', 'Q28A'), ('level', 'Q20A'), ('level', 'Q36A'), ('level', 'Q4A'),
('level', 'Q40A'), ('level', 'Q41A'), ('level', 'Q9A'), ('level', 'Q25A'),
('level', 'Q30A'), ('level', 'Q15A'), ('level', 'Q23A'), ('level', 'Q19A'),
('level', 'Q2A'), ('Q41A', 'Q19A'), ('Q28A', 'Q36A'), ('Q28A', 'Q40A'), ('Q28
A', 'Q2A'), ('Q28A', 'Q20A'), ('Q28A', 'Q9A'), ('Q40A', 'Q9A'), ('Q40A', 'Q30
A'), ('Q9A', 'Q30A'), ('Q30A', 'Q4A'), ('Q30A', 'Q36A'), ('Q36A', 'Q20A'),
('Q36A', 'Q19A'), ('Q36A', 'Q25A')]
```

In [174]:
```
import pyAgrum as gum
import pyAgrum.lib.notebook as gnb
anx_bn=gum.BayesNet('Anxiety')
anx_nodes = set()
for x,y in bayesian_network.edges():
    anx_nodes.add(x)
    anx_nodes.add(y)
for node in anx_nodes: anx_bn.add(node)
for link in bayesian_network.edges():
    anx_bn.addArc(*link)
anx_bn
```

Out[174]:



In [124]:
```
from pgmpy.inference import VariableElimination
from pgmpy.estimators import MaximumLikelihoodEstimator
```

In [175]:
```
mle = MaximumLikelihoodEstimator(bayesian_network, X_train)
bayesian_network.fit(X_train)
```

In [176]:
```
anx_inference = VariableElimination(bayesian_network)
```

In [163]:
```
def predict_level_proba(df_row, inference, levels = ['extremely severe', 'sever
    if 'level' in df_row.keys():
        del df_row['level']
    posterior_probs = anx_inference.query(variables=['level'], evidence=df_row)
    level_probas = -1
    max_proba_level = None
    for lvl in levels:
        proba = posterior_probs.get_value(level = lvl)
        if proba > level_probas:
            level_probas = proba
            max_proba_level = lvl
    return max_proba_level
```

In [188]:
```
total_x_train_len = len(X_train)
correct = 0
for row in range(total_x_train_len):
    pred_level = predict_level_proba(X_train.iloc[row].to_dict(),anx_inference)
    if pred_level == y_train.iloc[row]: correct += 1
print('train accuracy:', correct/total_x_train_len)
```

```
train accuracy: 0.9480829666876178
```

In [194]:
```
total_x_test_len = len(X_test)
correct_test = 0
for row in range(total_x_test_len):
    pred_level = predict_level_proba(X_test.iloc[row].to_dict(),anx_inference)
    if pred_level == y_test.iloc[row]: correct_test += 1
print('test accuracy:', correct_test/total_x_test_len)
```

```
test accuracy: 0.9329981143934633
```

References:

1.) Assessment of Anxiety, Depression and Stress using Machine Learning Models by Prince Kumar, Shruti Garg, Ashwani Garg (https://www.sciencedirect.com/science/article/pii/S1877050920311984)