```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from stellargraph import StellarGraph
from stellargraph.mapper import FullBatchNodeGenerator
from stellargraph.layer import GCN
from tensorflow.keras import layers, optimizers, losses, metrics, Model
from sklearn.metrics import accuracy_score

df = pd.read_csv("Biosensor_Stress_Analysis_Kaggle - Sheet1.csv")

for col in df.columns:
    if col != 'Stress Level':
        last_20 = df[col].tail(20)
        next_20 = np.arange(1, 21) * (last_20.mean() - last_20.iloc[-1]) +
last_20.iloc[-1]
        df = pd.concat([df, pd.DataFrame({col: next_20})], ignore_index=True)

stress_mapping = {'low': 0, 'medium': 1, 'high': 2}
df['Stress Level'] = df['Stress Level'].map(stress_mapping)

X = df.drop('Stress Level', axis=1)
y = df['Stress Level']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

graph = StellarGraph.from_networkx(df.corr().to_networkx())

generator = FullBatchNodeGenerator(graph)
train_gen = generator.flow(X_train_scaled.index, y_train)
test_gen = generator.flow(X_test_scaled.index, y_test)

gcn = GCN(layer_sizes=[16, 16], activations=["relu", "relu"],
generator=generator)
x_inp, x_out = gcn.in_out_tensors()
predictions = layers.Dense(units=len(stress_mapping.keys()),
activation="softmax")(x_out)

model = Model(inputs=x_inp, outputs=predictions)
```

```
model.compile(
    optimizer=optimizers.Adam(learning_rate=0.01),
    loss=losses.sparse_categorical_crossentropy,
    metrics=["acc"],
)

model.fit(train_gen, epochs=50, verbose=2, validation_data=test_gen)

Epoch 1/50
2/2 - 1s - loss: 1.0667 - acc: 0.4792 - val_loss: 1.0261 - val_acc: 0.6000 -
832ms/epoch - 416ms/step
Epoch 2/50
2/2 - 0s - loss: 0.9855 - acc: 0.6250 - val_loss: 0.9506 - val_acc: 0.6000 -
28ms/epoch - 14ms/step
...
Epoch 49/50
2/2 - 0s - loss: 0.1738 - acc: 0.9167 - val_loss: 0.2050 - val_acc: 0.8800 -
28ms/epoch - 14ms/step
Epoch 50/50
2/2 - 0s - loss: 0.1695 - acc: 0.9167 - val_loss: 0.2077 - val_acc: 0.8800 -
35ms/epoch - 18ms/step

test_metrics = model.evaluate(test_gen)
print("\nTest Set Metrics:")
for name, val in zip(model.metrics_names, test_metrics):
    print("\t{}: {:0.4f}".format(name, val))

1/1 - 0s - loss: 0.2077 - acc: 0.8800 - 127ms/epoch - 127ms/step

Test Set Metrics:
  loss: 0.2077
  acc: 0.8800

y_pred = model.predict(test_gen)
y_pred_classes = np.argmax(y_pred, axis=1)

accuracy = accuracy_score(y_test, y_pred_classes)
print(f"\nAccuracy: {accuracy:.4f}")

Accuracy: 0.8800
```