In [1]:

```
#Razat Siwakoti (A00046635)
#DMV302 - Assessment 2
#NN2.ipynb created on Jupyter notebook


#source: Scikit-learn(2015)
#https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.

#Deepika S. (2019)
#https://www.pluralsight.com/guides/machine-learning-neural-networks-scikit-learn
```

In [1]:

```
#importing necessary libraries
import pandas as pd
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_sc
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

In [2]:

```
# Load the training dataset
df_train = pd.read_csv("AtRiskStudentsTraining.csv")
df_train.head()
```

Out[2]:

	GPA	attendance	duration	language	at-risk
0	2.07	76	9586	41	1
1	1.97	19	3772	28	0
2	2.49	43	1506	10	1
3	1.94	82	9223	3	1
4	0.52	37	8232	64	0

In [3]:

```
# Load the training dataset
df_test = pd.read_csv("AtRiskStudentsTest.csv")
df_test.head()
```

Out[3]:

	GPA	attendance	duration	language	at-risk
0	0.94	29	2017	51	1
1	3.65	82	5722	74	1
2	2.41	69	4917	15	0
3	1.24	6	3720	50	0
4	2.14	54	2487	59	1

In [4]:

```
# Separate features and target variable for both training and test sets
X_train = df_train.drop('at-risk', axis=1)
y_train = df_train['at-risk']
X_test = df_test.drop('at-risk', axis=1)
y_test = df_test['at-risk']

# Initialize a list to store the results
results = []

# Test various configurations of hidden layers and neurons
```

```
hidden_layers_list = [1, 2, 3, 4, 5]   # You can extend this list as needed
neurons_list = [50, 60, 70, 80, 90, 100]  # You can extend this list as needed
```

In [5]:

```
for num_layers in hidden_layers_list:
    for num_neurons in neurons_list:
        # Create the neural network model
        model = MLPClassifier(hidden_layer_sizes=(num_neurons,) * num_layers, activation

        # Train the model on the training set
        model.fit(X_train, y_train)

        # Make predictions on the test set
        y_pred = model.predict(X_test)

        # Calculate the error using accuracy as the metric
        error = 1 - accuracy_score(y_test, y_pred)

        # Calculate and print metrics for the test set
        test_accuracy = accuracy_score(y_test, y_pred)
        test_precision = precision_score(y_test, y_pred)
        test_recall = recall_score(y_test, y_pred)
        test_f1 = f1_score(y_test, y_pred)

        # Store the results
        results.append({
            'Hidden Layers': num_layers,
            'Neurons per Layer': num_neurons,
            'Error on Test Set': error,
            'Test Accuracy' : test_accuracy,
            'Precision' : test_precision,
            'Recall' : test_recall,
            'F1 score' : test_f1,

        })
```

In [12]:

```
# Convert results to a DataFrame for better presentation
results_df = pd.DataFrame(results)
results_df
```

Out[12]:

	Hidden Layers	Neurons per Layer	Error on Test Set	Test Accuracy	Precision	Recall	F1 score
0	1	50	0.498	0.502	0.497959	0.987854	0.662144
1	1	60	0.494	0.506	0.500000	0.975709	0.661180
2	1	70	0.502	0.498	0.388889	0.028340	0.052830
3	1	80	0.484	0.516	0.567568	0.085020	0.147887
4	1	90	0.492	0.508	0.529412	0.036437	0.068182
5	1	100	0.502	0.498	0.357143	0.020243	0.038314
6	2	50	0.490	0.510	0.514286	0.145749	0.227129
7	2	60	0.486	0.514	0.666667	0.032389	0.061776
8	2	70	0.504	0.496	0.494990	1.000000	0.662198
9	2	80	0.492	0.508	0.666667	0.008097	0.016000
10	2	90	0.482	0.518	0.575000	0.093117	0.160279
11	2	100	0.500	0.500	0.428571	0.036437	0.067164
12	3	50	0.500	0.500	0.411765	0.028340	0.053030

13	3	60	0.498	0.502	0.473684	0.072874	0.126316
14	3	70	0.446	0.554	0.617647	0.255061	0.361032
15	3	80	0.482	0.518	0.714286	0.040486	0.076628
16	3	90	0.490	0.510	0.750000	0.012146	0.023904
17	3	100	0.496	0.504	0.428571	0.012146	0.023622
18	4	50	0.488	0.512	0.615385	0.032389	0.061538
19	4	60	0.496	0.504	0.498986	0.995951	0.664865
20	4	70	0.500	0.500	0.496970	0.995951	0.663073
21	4	80	0.496	0.504	0.498881	0.902834	0.642651
22	4	90	0.522	0.478	0.485417	0.943320	0.640990
23	4	100	0.510	0.490	0.277778	0.020243	0.037736
24	5	50	0.460	0.540	0.623188	0.174089	0.272152
25	5	60	0.492	0.508	0.512195	0.085020	0.145833
26	5	70	0.494	0.506	0.500000	0.995951	0.665765
27	5	80	0.498	0.502	0.444444	0.032389	0.060377
28	5	90	0.500	0.500	0.444444	0.048583	0.087591
29	5	100	0.498	0.502	0.497951	0.983806	0.661224

In [13]:

```
#sort best results based on 'error io test set' and print the first ten in table
best_configs = results_df.sort_values(by='Error on Test Set').head(10)
print("Top 10 Configurations:")
best_configs
```

```
Top 10 Configurations:
```

Out[13]:

	Hidden Layers	Neurons per Layer	Error on Test Set	Test Accuracy	Precision	Recall	F1 score
14	3	70	0.446	0.554	0.617647	0.255061	0.361032
24	5	50	0.460	0.540	0.623188	0.174089	0.272152
15	3	80	0.482	0.518	0.714286	0.040486	0.076628
10	2	90	0.482	0.518	0.575000	0.093117	0.160279
3	1	80	0.484	0.516	0.567568	0.085020	0.147887
7	2	60	0.486	0.514	0.666667	0.032389	0.061776
18	4	50	0.488	0.512	0.615385	0.032389	0.061538
6	2	50	0.490	0.510	0.514286	0.145749	0.227129
16	3	90	0.490	0.510	0.750000	0.012146	0.023904
25	5	60	0.492	0.508	0.512195	0.085020	0.145833