In [1]:
```
import pandas as pd

import numpy as np
import itertools
import category_encoders as ce

from numpy import mean
from numpy import std
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.feature_selection import f_classif
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import accuracy_score


from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, roc_auc_score, classification_report

import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
pip install pandas scikit-learn
```

```
Requirement already satisfied: pandas in c:\users\billi\anaconda3\lib\site-packages (2.0.3)
Requirement already satisfied: scikit-learn in c:\users\billi\anaconda3\lib\site-packages (1.3.0)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\billi\anaconda3\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\billi\anaconda3\lib\site-packages (from pandas) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\billi\anaconda3\lib\site-packages (from pandas) (2023.3)
Requirement already satisfied: numpy>=1.21.0 in c:\users\billi\anaconda3\lib\site-packages (from pandas) (1.24.3)
Requirement already satisfied: scipy>=1.5.0 in c:\users\billi\anaconda3\lib\site-packages (from scikit-learn) (1.11.1)
Requirement already satisfied: joblib>=1.1.1 in c:\users\billi\anaconda3\lib\site-packages (from scikit-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\billi\anaconda3\lib\site-packages (from scikit-learn) (2.2.0)
Requirement already satisfied: six>=1.5 in c:\users\billi\anaconda3\lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
import pandas as pd
import numpy as np
```

```
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import export_text
from sklearn.metrics import precision_score, recall_score, auc
from sklearn.metrics import roc_curve, accuracy_score, confusion_matrix
import seaborn as sns
from sklearn.metrics import ConfusionMatrixDisplay, classification_report
from sklearn.model_selection import train_test_split
import scipy.stats as stats
from scipy.stats import shapiro, normaltest
import category_encoders as ce
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
```

In [4]:

```
#df = pd.read_csv('bank updated.csv')            Old .csv with numeric pdays.

df = pd.read_csv('bank updated categories.csv')

col_names = ['age',
        'job',
        'martial',
        'education',
        'cred_in_default',
        'balance',
        'housing',
        'loan',
        'contact',
        'last_contact_day',
        'last_contact_month',
        'last_contact_dur',
        'num_of_contacts_during_campaign',
        'past_days',
        'prev_contacts',
        'prev_outcome',
        'sub_term_deposit']
df.columns = col_names
df.head()
```

Out[4]:

	age	job	martial	education	cred_in_default	balance	housing	loan	contact	last_contact_day	last_contact_month	last_contact_dur	num_
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	oct	79	
1	30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun	199	
2	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	may	226	
3	39	technician	married	secondary	no	147	yes	no	cellular	6	may	151	
4	41	entrepreneur	married	tertiary	no	221	yes	no	unknown	14	may	57	

In [5]:

```
##Prep for MLP
print("Bank data set dimensions : {}".format(df.shape))
```

Bank data set dimensions : (4521, 17)

In [7]:

```
from sklearn.preprocessing import OrdinalEncoder

# Load the dataset
df = pd.read_csv('bank updated categories.csv')

# Define features and target
X = df[['age', 'job', 'marital', 'education', 'balance', 'housing',
        'loan', 'contact', 'day', 'month', 'duration', 'campaign', 'pdays',
        'previous', 'poutcome']]
y = df['y']

# Define the columns to encode
cols_to_encode = ['job', 'marital', 'education', 'housing', 'loan', 'contact', 'month', 'pdays', 'poutcome']

# Filter cols_to_encode to include only those present in X
cols_to_encode = [col for col in cols_to_encode if col in X.columns]

# Extract column indices for encoding
col_indices = [X.columns.get_loc(col) for col in cols_to_encode]

# Create the encoder
```

```
encoder = OrdinalEncoder()
X_enc = X.copy()  # Make a copy of X
X_enc[cols_to_encode] = encoder.fit_transform(X[cols_to_encode])

# Splitting Data
X_train, X_test, y_train, y_test = train_test_split(X_enc, y, test_size=0.3, stratify=y, random_state=3)


print("y_test set dimensions : {}".format(y_test.shape))

# Define the MLPClassifier
mlp = MLPClassifier(
    max_iter=200,
    alpha=0.1,
    activation='logistic',
    solver='adam')

# Fit the model to the training data
mlp.fit(X_enc, y)
mlp_predict = mlp.predict(X_enc)
```

y_test set dimensions : (1357,)

In [36]:

```
###Part B question B
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report

# Define a range of number of neurons (k) and number of iterations
neurons_range = range(1, 26)  # From 1 to 25 neurons
iterations_range = [100, 200, 300, 400, 500] # Specify a range of iterations to test

# Define parameters for the grid search
param_grid = {'hidden_layer_sizes': [(k,) for k in neurons_range],  # Single hidden layer with k neurons
              'max_iter': iterations_range}  # Varying number of iterations

# Create an MLPClassifier instance
mlp = MLPClassifier()

# Perform grid search with cross-validation
grid_search = GridSearchCV(mlp, param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid_search.fit(X_train, y_train)

# Get the best parameters and accuracy
```

```
best_params = grid_search.best_params_
best_accuracy = grid_search.best_score_

# Get the best iteration for the best parameters
best_iter = grid_search.cv_results_['param_max_iter'][grid_search.best_index_]

# Predict on the test set using the best parameters
y_pred = grid_search.best_estimator_.predict(X_test)

# Generate and print the classification report
print("Classification Report:")
print(classification_report(y_test, y_pred))

print("Best parameters:", best_params)
print("Best accuracy: {:.2f}%".format(best_accuracy * 100))
print("Best iteration:", best_iter)
```

```
Classification Report:
              precision    recall  f1-score   support

          no       0.89      1.00      0.94      1201
         yes       0.00      0.00      0.00       156

    accuracy                           0.89      1357
   macro avg       0.44      0.50      0.47      1357
weighted avg       0.78      0.89      0.83      1357

Best parameters: {'hidden_layer_sizes': (3,), 'max_iter': 300}
Best accuracy: 88.72%
Best iteration: 300
```

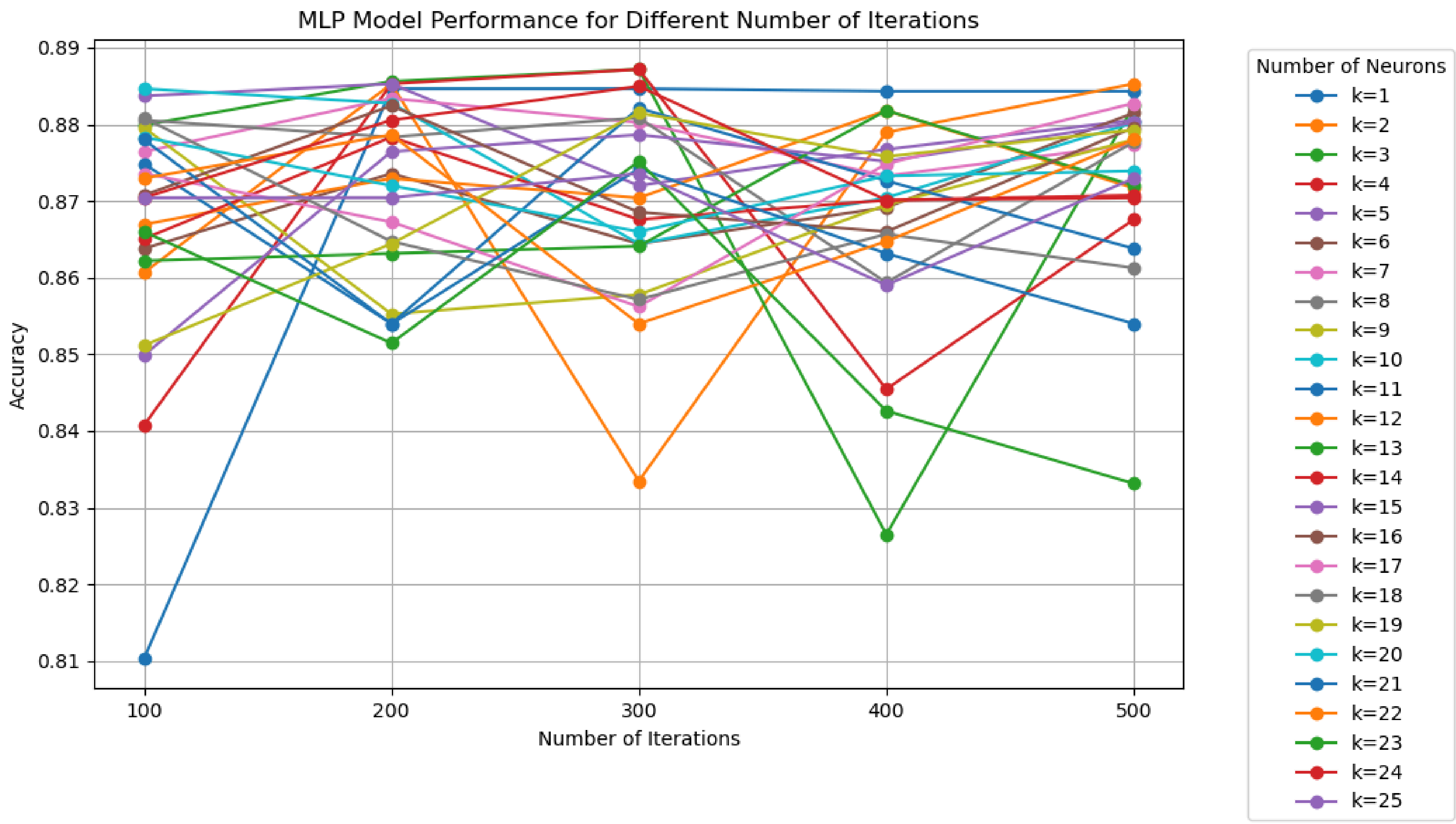
In [37]:

```
####Part B question B -  Plotting the accuracy for each number of iterations
results = grid_search.cv_results_
mean_test_scores = np.array([results['mean_test_score'][i::len(iterations_range)] for i in range(len(iterations_range))])

plt.figure(figsize=(10, 6))
for i, k in enumerate(neurons_range):
    plt.plot(iterations_range, mean_test_scores[:, i], marker='o', label=f'k={k}')

plt.title('MLP Model Performance for Different Number of Iterations')
plt.xlabel('Number of Iterations')
plt.ylabel('Accuracy')
plt.xticks(iterations_range)
plt.legend(title='Number of Neurons', bbox_to_anchor=(1.05, 1), loc='upper left')
```

```
plt.grid(True)
plt.show()
```



In [12]:
```
##Question c Part B
from sklearn.metrics import log_loss
from sklearn.neural_network import MLPClassifier
import matplotlib.pyplot as plt

# Initialize variables to store the losses for each iteration
train_losses = []
```

```
test_losses = []

# Set the maximum number of iterations
max_iter = 500

# Create an MLPClassifier instance with warm_start=True to enable incremental fitting
mlp = MLPClassifier(max_iter=1, warm_start=True, verbose=False, random_state=42)

for i in range(max_iter):
    # Fit the model to the training data for one iteration
    mlp.fit(X_train, y_train)

    # Append the current training loss
    train_losses.append(mlp.loss_)

    # Calculate the loss on the test set
    y_pred_prob = mlp.predict_proba(X_test)
    test_loss = log_loss(y_test, y_pred_prob)  # Using log_loss function from scikit-learn
    test_losses.append(test_loss)

    # Print the loss for the current iteration
    print(f'Iteration {i+1}: Training Loss = {train_losses[-1]:.4f}, Test Loss = {test_losses[-1]:.4f}')

# Plotting the loss curves
plt.figure(figsize=(10, 6))
plt.plot(train_losses, label='Training Loss')
plt.plot(test_losses, label='Test Loss')
plt.title('MLP Loss Curve')
plt.xlabel('Iterations')
plt.ylabel('Loss')
plt.legend()
plt.grid(True)
plt.show()

# Predict using the testing set after all iterations
mlp_predict = mlp.predict(X_test)

# Calculate MLP Accuracy
mlp_accuracy = accuracy_score(y_test, mlp_predict)

# MLP Classification report
mlp_classification_report = classification_report(y_test, mlp_predict)

# MLP Training set score
```

```
mlp_training_score = mlp.score(X_train, y_train)

# MLP Testing set score
mlp_testing_score = mlp.score(X_test, y_test)

# Print the accuracy and classification report
print(f'MLP Accuracy: {mlp_accuracy:.2%}')
print('MLP Classification Report:')
print(mlp_classification_report)
print(f'MLP Training set score: {mlp_training_score:.2%}')
print(f'MLP Testing set score: {mlp_testing_score:.2%}')
```

```
Iteration 1: Training Loss = 13.4876, Test Loss = 4.0516
Iteration 2: Training Loss = 4.1469, Test Loss = 5.0731
Iteration 3: Training Loss = 2.5687, Test Loss = 1.3078
Iteration 4: Training Loss = 2.0093, Test Loss = 1.5225
Iteration 5: Training Loss = 2.1275, Test Loss = 0.9143
Iteration 6: Training Loss = 2.3489, Test Loss = 2.4190
Iteration 7: Training Loss = 2.5962, Test Loss = 2.3637
Iteration 8: Training Loss = 1.3094, Test Loss = 0.9191
Iteration 9: Training Loss = 2.5741, Test Loss = 1.9250
Iteration 10: Training Loss = 1.6631, Test Loss = 0.8318
Iteration 11: Training Loss = 1.7047, Test Loss = 1.3988
Iteration 12: Training Loss = 2.0125, Test Loss = 0.9752
Iteration 13: Training Loss = 2.1963, Test Loss = 2.3309
Iteration 14: Training Loss = 2.1530, Test Loss = 1.8261
Iteration 15: Training Loss = 1.5882, Test Loss = 1.1291
Iteration 16: Training Loss = 1.9510, Test Loss = 0.9964
Iteration 17: Training Loss = 1.9929, Test Loss = 1.6724
Iteration 18: Training Loss = 1.4802, Test Loss = 0.8100
Iteration 19: Training Loss = 2.0443, Test Loss = 0.9573
Iteration 20: Training Loss = 1.8737, Test Loss = 1.4050
Iteration 21: Training Loss = 1.7824, Test Loss = 0.9115
Iteration 22: Training Loss = 1.3435, Test Loss = 1.0898
Iteration 23: Training Loss = 1.8807, Test Loss = 0.9867
Iteration 24: Training Loss = 1.8161, Test Loss = 1.2158
Iteration 25: Training Loss = 1.7088, Test Loss = 0.9333
Iteration 26: Training Loss = 1.0554, Test Loss = 0.9265
Iteration 27: Training Loss = 2.0154, Test Loss = 0.8842
Iteration 28: Training Loss = 1.9139, Test Loss = 1.5598
Iteration 29: Training Loss = 1.4717, Test Loss = 1.0612
Iteration 30: Training Loss = 1.5068, Test Loss = 0.8030
Iteration 31: Training Loss = 2.0029, Test Loss = 1.9938
Iteration 32: Training Loss = 1.4369, Test Loss = 0.8794
Iteration 33: Training Loss = 2.0797, Test Loss = 0.9779
Iteration 34: Training Loss = 1.4503, Test Loss = 0.8020
Iteration 35: Training Loss = 1.7611, Test Loss = 1.1759
Iteration 36: Training Loss = 1.8272, Test Loss = 0.8647
Iteration 37: Training Loss = 1.1258, Test Loss = 1.0587
Iteration 38: Training Loss = 1.8488, Test Loss = 0.8470
Iteration 39: Training Loss = 1.1521, Test Loss = 1.2245
Iteration 40: Training Loss = 1.7702, Test Loss = 0.9009
Iteration 41: Training Loss = 1.2777, Test Loss = 1.3327
Iteration 42: Training Loss = 1.6640, Test Loss = 0.9970
Iteration 43: Training Loss = 1.4731, Test Loss = 1.1991
Iteration 44: Training Loss = 1.7295, Test Loss = 0.9257
```

```
Iteration 45: Training Loss = 1.1390, Test Loss = 1.2110
Iteration 46: Training Loss = 1.7357, Test Loss = 0.9030
Iteration 47: Training Loss = 1.1244, Test Loss = 1.2244
Iteration 48: Training Loss = 1.7038, Test Loss = 0.9263
Iteration 49: Training Loss = 1.1415, Test Loss = 1.1722
Iteration 50: Training Loss = 1.7422, Test Loss = 0.8945
Iteration 51: Training Loss = 1.0821, Test Loss = 1.1023
Iteration 52: Training Loss = 1.7959, Test Loss = 0.8990
Iteration 53: Training Loss = 1.6452, Test Loss = 1.0365
Iteration 54: Training Loss = 1.2996, Test Loss = 0.7142
Iteration 55: Training Loss = 1.5694, Test Loss = 0.9666
Iteration 56: Training Loss = 1.0229, Test Loss = 1.3832
Iteration 57: Training Loss = 1.6779, Test Loss = 1.0632
Iteration 58: Training Loss = 1.5142, Test Loss = 1.3027
Iteration 59: Training Loss = 1.6389, Test Loss = 1.1669
Iteration 60: Training Loss = 1.3424, Test Loss = 1.0290
Iteration 61: Training Loss = 1.6390, Test Loss = 1.1549
Iteration 62: Training Loss = 1.5301, Test Loss = 1.0875
Iteration 63: Training Loss = 1.6571, Test Loss = 1.0969
Iteration 64: Training Loss = 1.4036, Test Loss = 1.0652
Iteration 65: Training Loss = 1.8671, Test Loss = 0.9576
Iteration 66: Training Loss = 1.5980, Test Loss = 0.9079
Iteration 67: Training Loss = 1.5297, Test Loss = 0.9445
Iteration 68: Training Loss = 1.5507, Test Loss = 0.9588
Iteration 69: Training Loss = 1.5320, Test Loss = 0.9652
Iteration 70: Training Loss = 1.5513, Test Loss = 0.9761
Iteration 71: Training Loss = 1.5141, Test Loss = 0.9734
Iteration 72: Training Loss = 1.6021, Test Loss = 1.0425
Iteration 73: Training Loss = 1.2170, Test Loss = 0.6914
Iteration 74: Training Loss = 0.9840, Test Loss = 1.2667
Iteration 75: Training Loss = 1.6797, Test Loss = 0.9529
Iteration 76: Training Loss = 1.5274, Test Loss = 1.2810
Iteration 77: Training Loss = 1.5074, Test Loss = 0.9087
Iteration 78: Training Loss = 1.2549, Test Loss = 0.9133
Iteration 79: Training Loss = 1.9103, Test Loss = 0.9189
Iteration 80: Training Loss = 1.5102, Test Loss = 0.9363
Iteration 81: Training Loss = 1.6618, Test Loss = 1.0778
Iteration 82: Training Loss = 1.6290, Test Loss = 0.8338
Iteration 83: Training Loss = 1.0404, Test Loss = 1.1089
Iteration 84: Training Loss = 1.6822, Test Loss = 0.9090
Iteration 85: Training Loss = 1.2782, Test Loss = 1.2000
Iteration 86: Training Loss = 1.6060, Test Loss = 0.9938
Iteration 87: Training Loss = 1.3700, Test Loss = 1.1893
Iteration 88: Training Loss = 1.6075, Test Loss = 1.0011
```

```
Iteration 89: Training Loss = 1.2784, Test Loss = 1.2026
Iteration 90: Training Loss = 1.5957, Test Loss = 1.0283
Iteration 91: Training Loss = 1.3803, Test Loss = 0.9164
Iteration 92: Training Loss = 1.7278, Test Loss = 0.8981
Iteration 93: Training Loss = 1.3085, Test Loss = 1.0714
Iteration 94: Training Loss = 1.6808, Test Loss = 0.9495
Iteration 95: Training Loss = 1.1434, Test Loss = 1.3160
Iteration 96: Training Loss = 1.4981, Test Loss = 1.0263
Iteration 97: Training Loss = 1.5498, Test Loss = 0.7461
Iteration 98: Training Loss = 1.0157, Test Loss = 1.2220
Iteration 99: Training Loss = 1.6524, Test Loss = 1.0398
Iteration 100: Training Loss = 1.5346, Test Loss = 0.8632
Iteration 101: Training Loss = 1.6613, Test Loss = 1.0961
Iteration 102: Training Loss = 1.5828, Test Loss = 0.8486
Iteration 103: Training Loss = 1.4259, Test Loss = 0.6671
Iteration 104: Training Loss = 1.0984, Test Loss = 1.1484
Iteration 105: Training Loss = 1.6687, Test Loss = 0.9751
Iteration 106: Training Loss = 1.5118, Test Loss = 0.7813
Iteration 107: Training Loss = 1.3887, Test Loss = 0.9141
Iteration 108: Training Loss = 1.6326, Test Loss = 1.0262
Iteration 109: Training Loss = 1.6451, Test Loss = 0.8553
Iteration 110: Training Loss = 1.0229, Test Loss = 1.0614
Iteration 111: Training Loss = 1.6961, Test Loss = 0.8747
Iteration 112: Training Loss = 1.0164, Test Loss = 1.0715
Iteration 113: Training Loss = 1.7110, Test Loss = 0.8623
Iteration 114: Training Loss = 1.0165, Test Loss = 1.0970
Iteration 115: Training Loss = 1.6905, Test Loss = 0.8753
Iteration 116: Training Loss = 1.0190, Test Loss = 1.0474
Iteration 117: Training Loss = 1.7063, Test Loss = 0.8634
Iteration 118: Training Loss = 1.0315, Test Loss = 1.1390
Iteration 119: Training Loss = 1.6560, Test Loss = 0.9327
Iteration 120: Training Loss = 1.5201, Test Loss = 0.9888
Iteration 121: Training Loss = 1.3030, Test Loss = 0.7609
Iteration 122: Training Loss = 1.6382, Test Loss = 1.0263
Iteration 123: Training Loss = 1.6442, Test Loss = 0.8646
Iteration 124: Training Loss = 0.9975, Test Loss = 0.9599
Iteration 125: Training Loss = 1.7255, Test Loss = 0.8646
Iteration 126: Training Loss = 1.0425, Test Loss = 1.1853
Iteration 127: Training Loss = 1.6315, Test Loss = 0.9689
Iteration 128: Training Loss = 1.1612, Test Loss = 1.2193
Iteration 129: Training Loss = 1.5931, Test Loss = 0.9984
Iteration 130: Training Loss = 1.3272, Test Loss = 1.0594
Iteration 131: Training Loss = 1.6436, Test Loss = 0.9313
Iteration 132: Training Loss = 1.0317, Test Loss = 1.1240
```

Iteration 133: Training Loss = 1.6681, Test Loss = 0.9055
Iteration 134: Training Loss = 1.0231, Test Loss = 1.1185
Iteration 135: Training Loss = 1.6702, Test Loss = 0.8984
Iteration 136: Training Loss = 1.0306, Test Loss = 1.1269
Iteration 137: Training Loss = 1.6588, Test Loss = 0.9117
Iteration 138: Training Loss = 1.5043, Test Loss = 1.0470
Iteration 139: Training Loss = 1.2140, Test Loss = 0.6966
Iteration 140: Training Loss = 2.0152, Test Loss = 0.9470
Iteration 141: Training Loss = 0.7641, Test Loss = 1.2540
Iteration 142: Training Loss = 1.6991, Test Loss = 0.8703
Iteration 143: Training Loss = 1.2120, Test Loss = 1.2203
Iteration 144: Training Loss = 1.5984, Test Loss = 0.9525
Iteration 145: Training Loss = 1.2536, Test Loss = 1.1932
Iteration 146: Training Loss = 1.6040, Test Loss = 0.9611
Iteration 147: Training Loss = 1.4949, Test Loss = 1.0513
Iteration 148: Training Loss = 1.4114, Test Loss = 0.8397
Iteration 149: Training Loss = 1.6039, Test Loss = 1.0057
Iteration 150: Training Loss = 1.2116, Test Loss = 0.7676
Iteration 151: Training Loss = 1.6241, Test Loss = 1.0159
Iteration 152: Training Loss = 1.3435, Test Loss = 0.9772
Iteration 153: Training Loss = 1.8214, Test Loss = 0.9239
Iteration 154: Training Loss = 1.4884, Test Loss = 0.8169
Iteration 155: Training Loss = 1.3635, Test Loss = 0.9935
Iteration 156: Training Loss = 1.4798, Test Loss = 0.7021
Iteration 157: Training Loss = 1.3538, Test Loss = 0.9103
Iteration 158: Training Loss = 1.5438, Test Loss = 0.9035
Iteration 159: Training Loss = 1.2256, Test Loss = 0.6549
Iteration 160: Training Loss = 1.2805, Test Loss = 0.6427
Iteration 161: Training Loss = 1.2754, Test Loss = 0.7035
Iteration 162: Training Loss = 1.8884, Test Loss = 0.8282
Iteration 163: Training Loss = 1.3239, Test Loss = 0.8178
Iteration 164: Training Loss = 1.8287, Test Loss = 0.8139
Iteration 165: Training Loss = 1.4242, Test Loss = 0.9614
Iteration 166: Training Loss = 1.3202, Test Loss = 0.6886
Iteration 167: Training Loss = 0.9672, Test Loss = 1.2270
Iteration 168: Training Loss = 1.5818, Test Loss = 1.0102
Iteration 169: Training Loss = 1.1103, Test Loss = 1.1576
Iteration 170: Training Loss = 1.5988, Test Loss = 0.9237
Iteration 171: Training Loss = 1.1454, Test Loss = 1.1981
Iteration 172: Training Loss = 1.5547, Test Loss = 0.9761
Iteration 173: Training Loss = 1.3326, Test Loss = 0.9238
Iteration 174: Training Loss = 1.6753, Test Loss = 0.8335
Iteration 175: Training Loss = 1.4712, Test Loss = 0.9847
Iteration 176: Training Loss = 1.5850, Test Loss = 1.0276

```
Iteration 177: Training Loss = 1.2113, Test Loss = 0.6568
Iteration 178: Training Loss = 1.3935, Test Loss = 0.8059
Iteration 179: Training Loss = 1.5682, Test Loss = 0.9451
Iteration 180: Training Loss = 1.1722, Test Loss = 0.6848
Iteration 181: Training Loss = 1.4457, Test Loss = 1.0798
Iteration 182: Training Loss = 1.3215, Test Loss = 0.9125
Iteration 183: Training Loss = 1.3974, Test Loss = 0.7624
Iteration 184: Training Loss = 1.5254, Test Loss = 0.9070
Iteration 185: Training Loss = 1.2040, Test Loss = 0.6536
Iteration 186: Training Loss = 1.1475, Test Loss = 0.7720
Iteration 187: Training Loss = 1.8446, Test Loss = 0.7963
Iteration 188: Training Loss = 1.4126, Test Loss = 0.9000
Iteration 189: Training Loss = 1.5068, Test Loss = 0.8841
Iteration 190: Training Loss = 1.1742, Test Loss = 0.7706
Iteration 191: Training Loss = 1.3551, Test Loss = 0.8571
Iteration 192: Training Loss = 1.5437, Test Loss = 0.9184
Iteration 193: Training Loss = 1.1985, Test Loss = 0.6418
Iteration 194: Training Loss = 1.0649, Test Loss = 1.0657
Iteration 195: Training Loss = 1.5920, Test Loss = 0.9523
Iteration 196: Training Loss = 1.1499, Test Loss = 1.1550
Iteration 197: Training Loss = 1.5572, Test Loss = 0.9590
Iteration 198: Training Loss = 1.2600, Test Loss = 1.0791
Iteration 199: Training Loss = 1.5862, Test Loss = 0.9002
Iteration 200: Training Loss = 1.0826, Test Loss = 1.2516
Iteration 201: Training Loss = 1.4605, Test Loss = 1.0111
Iteration 202: Training Loss = 1.1926, Test Loss = 1.1205
Iteration 203: Training Loss = 1.5790, Test Loss = 0.9182
Iteration 204: Training Loss = 1.0783, Test Loss = 1.2715
Iteration 205: Training Loss = 1.4734, Test Loss = 0.9958
Iteration 206: Training Loss = 1.2212, Test Loss = 1.0478
Iteration 207: Training Loss = 1.6110, Test Loss = 0.8714
Iteration 208: Training Loss = 1.0741, Test Loss = 1.2476
Iteration 209: Training Loss = 1.5071, Test Loss = 1.1052
Iteration 210: Training Loss = 1.0789, Test Loss = 1.2093
Iteration 211: Training Loss = 1.5764, Test Loss = 0.9396
Iteration 212: Training Loss = 1.2712, Test Loss = 1.0373
Iteration 213: Training Loss = 1.6108, Test Loss = 0.9028
Iteration 214: Training Loss = 1.0188, Test Loss = 1.1421
Iteration 215: Training Loss = 1.5785, Test Loss = 0.9505
Iteration 216: Training Loss = 1.2302, Test Loss = 1.1291
Iteration 217: Training Loss = 1.5737, Test Loss = 0.9540
Iteration 218: Training Loss = 1.3143, Test Loss = 0.8208
Iteration 219: Training Loss = 1.7394, Test Loss = 0.8263
Iteration 220: Training Loss = 1.5614, Test Loss = 1.0146
```

```
Iteration 221: Training Loss = 1.2554, Test Loss = 0.8797
Iteration 222: Training Loss = 1.5921, Test Loss = 1.0097
Iteration 223: Training Loss = 1.3927, Test Loss = 0.8824
Iteration 224: Training Loss = 1.7144, Test Loss = 0.9085
Iteration 225: Training Loss = 1.5762, Test Loss = 0.9830
Iteration 226: Training Loss = 1.2196, Test Loss = 0.8406
Iteration 227: Training Loss = 1.5680, Test Loss = 0.9485
Iteration 228: Training Loss = 1.2206, Test Loss = 0.9857
Iteration 229: Training Loss = 1.8182, Test Loss = 0.8377
Iteration 230: Training Loss = 1.5011, Test Loss = 0.8484
Iteration 231: Training Loss = 1.5502, Test Loss = 0.9631
Iteration 232: Training Loss = 1.1992, Test Loss = 0.8120
Iteration 233: Training Loss = 1.5255, Test Loss = 0.8698
Iteration 234: Training Loss = 1.1788, Test Loss = 0.6081
Iteration 235: Training Loss = 1.3063, Test Loss = 0.6449
Iteration 236: Training Loss = 0.9217, Test Loss = 1.1302
Iteration 237: Training Loss = 1.5646, Test Loss = 0.9580
Iteration 238: Training Loss = 1.0566, Test Loss = 1.1570
Iteration 239: Training Loss = 1.5649, Test Loss = 0.9143
Iteration 240: Training Loss = 1.3029, Test Loss = 0.8817
Iteration 241: Training Loss = 1.6336, Test Loss = 0.8246
Iteration 242: Training Loss = 1.4329, Test Loss = 0.9936
Iteration 243: Training Loss = 1.5752, Test Loss = 0.9858
Iteration 244: Training Loss = 1.1627, Test Loss = 0.6001
Iteration 245: Training Loss = 1.2777, Test Loss = 0.9314
Iteration 246: Training Loss = 1.3002, Test Loss = 0.5909
Iteration 247: Training Loss = 1.3727, Test Loss = 0.8349
Iteration 248: Training Loss = 1.5012, Test Loss = 0.8266
Iteration 249: Training Loss = 1.4594, Test Loss = 0.7943
Iteration 250: Training Loss = 1.5406, Test Loss = 0.8732
Iteration 251: Training Loss = 1.1442, Test Loss = 0.7514
Iteration 252: Training Loss = 0.9103, Test Loss = 0.9145
Iteration 253: Training Loss = 1.6664, Test Loss = 0.7868
Iteration 254: Training Loss = 1.4514, Test Loss = 0.9235
Iteration 255: Training Loss = 1.5886, Test Loss = 0.9898
Iteration 256: Training Loss = 1.1851, Test Loss = 0.6229
Iteration 257: Training Loss = 0.9514, Test Loss = 1.1320
Iteration 258: Training Loss = 1.5664, Test Loss = 0.9261
Iteration 259: Training Loss = 1.1658, Test Loss = 1.1243
Iteration 260: Training Loss = 1.5099, Test Loss = 0.9498
Iteration 261: Training Loss = 1.1555, Test Loss = 1.1291
Iteration 262: Training Loss = 1.5475, Test Loss = 0.9240
Iteration 263: Training Loss = 1.2983, Test Loss = 0.7841
Iteration 264: Training Loss = 1.6998, Test Loss = 0.7956
```

Iteration 265: Training Loss = 1.5254, Test Loss = 0.9235
Iteration 266: Training Loss = 1.3151, Test Loss = 0.7777
Iteration 267: Training Loss = 1.5533, Test Loss = 0.9134
Iteration 268: Training Loss = 1.2021, Test Loss = 0.6478
Iteration 269: Training Loss = 0.9437, Test Loss = 1.1731
Iteration 270: Training Loss = 1.5250, Test Loss = 0.9709
Iteration 271: Training Loss = 0.9857, Test Loss = 1.0541
Iteration 272: Training Loss = 1.6042, Test Loss = 0.8427
Iteration 273: Training Loss = 1.0499, Test Loss = 1.1725
Iteration 274: Training Loss = 1.5110, Test Loss = 1.0042
Iteration 275: Training Loss = 1.0065, Test Loss = 1.1496
Iteration 276: Training Loss = 1.5489, Test Loss = 0.9088
Iteration 277: Training Loss = 1.2924, Test Loss = 0.7836
Iteration 278: Training Loss = 1.7489, Test Loss = 0.8150
Iteration 279: Training Loss = 1.5348, Test Loss = 0.9573
Iteration 280: Training Loss = 1.2360, Test Loss = 0.7760
Iteration 281: Training Loss = 1.5864, Test Loss = 0.9706
Iteration 282: Training Loss = 1.2378, Test Loss = 0.9347
Iteration 283: Training Loss = 1.7584, Test Loss = 0.8715
Iteration 284: Training Loss = 1.4039, Test Loss = 0.9457
Iteration 285: Training Loss = 1.3337, Test Loss = 0.9037
Iteration 286: Training Loss = 1.3715, Test Loss = 0.8245
Iteration 287: Training Loss = 1.5097, Test Loss = 0.8574
Iteration 288: Training Loss = 1.1997, Test Loss = 0.7531
Iteration 289: Training Loss = 1.5770, Test Loss = 0.9477
Iteration 290: Training Loss = 1.4603, Test Loss = 0.7855
Iteration 291: Training Loss = 1.0022, Test Loss = 1.1513
Iteration 292: Training Loss = 1.4759, Test Loss = 1.0969
Iteration 293: Training Loss = 1.0023, Test Loss = 1.1344
Iteration 294: Training Loss = 1.5530, Test Loss = 0.9294
Iteration 295: Training Loss = 1.2408, Test Loss = 0.9720
Iteration 296: Training Loss = 1.5695, Test Loss = 0.9009
Iteration 297: Training Loss = 1.0145, Test Loss = 1.1225
Iteration 298: Training Loss = 1.5251, Test Loss = 0.9930
Iteration 299: Training Loss = 1.0054, Test Loss = 1.1189
Iteration 300: Training Loss = 1.5673, Test Loss = 0.9131
Iteration 301: Training Loss = 1.1799, Test Loss = 1.1152
Iteration 302: Training Loss = 1.4826, Test Loss = 1.0896
Iteration 303: Training Loss = 1.0974, Test Loss = 1.2112
Iteration 304: Training Loss = 1.4966, Test Loss = 1.0324
Iteration 305: Training Loss = 1.0213, Test Loss = 1.1854
Iteration 306: Training Loss = 1.5337, Test Loss = 0.9704
Iteration 307: Training Loss = 1.0308, Test Loss = 1.1242
Iteration 308: Training Loss = 1.5490, Test Loss = 0.9404

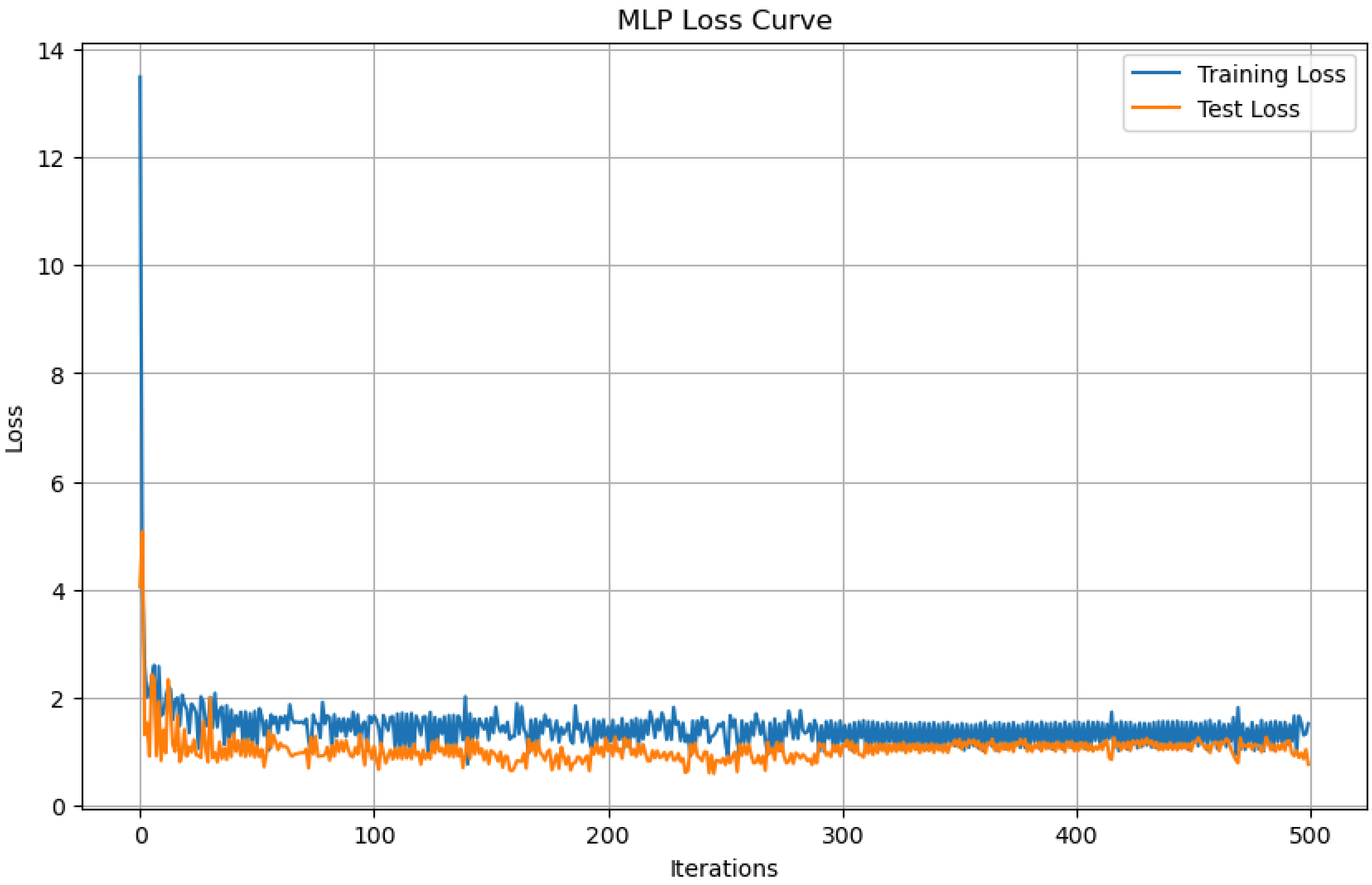
```
Iteration 309: Training Loss = 1.2544, Test Loss = 0.9370
Iteration 310: Training Loss = 1.5723, Test Loss = 0.8938
Iteration 311: Training Loss = 0.9990, Test Loss = 1.1518
Iteration 312: Training Loss = 1.5508, Test Loss = 0.9632
Iteration 313: Training Loss = 1.0063, Test Loss = 1.1300
Iteration 314: Training Loss = 1.5513, Test Loss = 0.9377
Iteration 315: Training Loss = 1.1486, Test Loss = 1.1352
Iteration 316: Training Loss = 1.5328, Test Loss = 0.9728
Iteration 317: Training Loss = 1.0525, Test Loss = 1.1425
Iteration 318: Training Loss = 1.5078, Test Loss = 1.0170
Iteration 319: Training Loss = 1.0297, Test Loss = 1.1409
Iteration 320: Training Loss = 1.5408, Test Loss = 0.9530
Iteration 321: Training Loss = 1.1754, Test Loss = 1.1278
Iteration 322: Training Loss = 1.5229, Test Loss = 0.9995
Iteration 323: Training Loss = 1.0579, Test Loss = 1.1400
Iteration 324: Training Loss = 1.5220, Test Loss = 0.9965
Iteration 325: Training Loss = 1.0523, Test Loss = 1.1451
Iteration 326: Training Loss = 1.5063, Test Loss = 1.0193
Iteration 327: Training Loss = 1.0339, Test Loss = 1.1724
Iteration 328: Training Loss = 1.5338, Test Loss = 0.9743
Iteration 329: Training Loss = 1.0712, Test Loss = 1.1277
Iteration 330: Training Loss = 1.5175, Test Loss = 1.0002
Iteration 331: Training Loss = 1.0774, Test Loss = 1.1377
Iteration 332: Training Loss = 1.5049, Test Loss = 1.0325
Iteration 333: Training Loss = 1.0409, Test Loss = 1.1511
Iteration 334: Training Loss = 1.5379, Test Loss = 0.9743
Iteration 335: Training Loss = 1.1021, Test Loss = 1.1262
Iteration 336: Training Loss = 1.5245, Test Loss = 1.0089
Iteration 337: Training Loss = 1.0389, Test Loss = 1.1327
Iteration 338: Training Loss = 1.5180, Test Loss = 1.0116
Iteration 339: Training Loss = 1.0418, Test Loss = 1.1456
Iteration 340: Training Loss = 1.5361, Test Loss = 0.9789
Iteration 341: Training Loss = 1.0751, Test Loss = 1.1288
Iteration 342: Training Loss = 1.5000, Test Loss = 1.0281
Iteration 343: Training Loss = 1.0319, Test Loss = 1.1179
Iteration 344: Training Loss = 1.5271, Test Loss = 0.9897
Iteration 345: Training Loss = 1.0880, Test Loss = 1.1311
Iteration 346: Training Loss = 1.5390, Test Loss = 0.9793
Iteration 347: Training Loss = 1.1299, Test Loss = 1.1423
Iteration 348: Training Loss = 1.4832, Test Loss = 1.1184
Iteration 349: Training Loss = 1.1034, Test Loss = 1.2128
Iteration 350: Training Loss = 1.4957, Test Loss = 1.0909
Iteration 351: Training Loss = 1.0815, Test Loss = 1.2241
Iteration 352: Training Loss = 1.5191, Test Loss = 1.0416
```

```
Iteration 353: Training Loss = 1.0214, Test Loss = 1.1764
Iteration 354: Training Loss = 1.5024, Test Loss = 1.0960
Iteration 355: Training Loss = 1.0732, Test Loss = 1.2178
Iteration 356: Training Loss = 1.4878, Test Loss = 1.1387
Iteration 357: Training Loss = 1.1076, Test Loss = 1.2148
Iteration 358: Training Loss = 1.4978, Test Loss = 1.0905
Iteration 359: Training Loss = 1.0627, Test Loss = 1.1947
Iteration 360: Training Loss = 1.5258, Test Loss = 1.0295
Iteration 361: Training Loss = 1.0142, Test Loss = 1.1409
Iteration 362: Training Loss = 1.5537, Test Loss = 0.9676
Iteration 363: Training Loss = 1.1356, Test Loss = 1.1540
Iteration 364: Training Loss = 1.4837, Test Loss = 1.1248
Iteration 365: Training Loss = 1.0904, Test Loss = 1.2314
Iteration 366: Training Loss = 1.5287, Test Loss = 1.0204
Iteration 367: Training Loss = 1.0180, Test Loss = 1.1281
Iteration 368: Training Loss = 1.5343, Test Loss = 0.9949
Iteration 369: Training Loss = 1.1473, Test Loss = 1.1556
Iteration 370: Training Loss = 1.5062, Test Loss = 1.0363
Iteration 371: Training Loss = 1.0517, Test Loss = 1.1509
Iteration 372: Training Loss = 1.5165, Test Loss = 1.0208
Iteration 373: Training Loss = 1.0895, Test Loss = 1.1462
Iteration 374: Training Loss = 1.5367, Test Loss = 0.9932
Iteration 375: Training Loss = 1.1351, Test Loss = 1.1685
Iteration 376: Training Loss = 1.4853, Test Loss = 1.1126
Iteration 377: Training Loss = 1.0714, Test Loss = 1.2271
Iteration 378: Training Loss = 1.5020, Test Loss = 1.1138
Iteration 379: Training Loss = 1.0685, Test Loss = 1.2239
Iteration 380: Training Loss = 1.5448, Test Loss = 0.9977
Iteration 381: Training Loss = 1.0458, Test Loss = 1.1216
Iteration 382: Training Loss = 1.5404, Test Loss = 0.9952
Iteration 383: Training Loss = 1.1822, Test Loss = 1.1385
Iteration 384: Training Loss = 1.5011, Test Loss = 1.0667
Iteration 385: Training Loss = 1.0533, Test Loss = 1.1621
Iteration 386: Training Loss = 1.5480, Test Loss = 1.0028
Iteration 387: Training Loss = 1.1259, Test Loss = 1.1759
Iteration 388: Training Loss = 1.5179, Test Loss = 1.0488
Iteration 389: Training Loss = 1.0655, Test Loss = 1.1509
Iteration 390: Training Loss = 1.5221, Test Loss = 1.0486
Iteration 391: Training Loss = 1.0395, Test Loss = 1.1494
Iteration 392: Training Loss = 1.5380, Test Loss = 1.0016
Iteration 393: Training Loss = 1.1568, Test Loss = 1.1751
Iteration 394: Training Loss = 1.5120, Test Loss = 1.0532
Iteration 395: Training Loss = 1.0461, Test Loss = 1.1412
Iteration 396: Training Loss = 1.5477, Test Loss = 0.9900
```

```
Iteration 397: Training Loss = 1.1683, Test Loss = 1.1771
Iteration 398: Training Loss = 1.5039, Test Loss = 1.0755
Iteration 399: Training Loss = 1.0719, Test Loss = 1.1878
Iteration 400: Training Loss = 1.5505, Test Loss = 1.0074
Iteration 401: Training Loss = 1.0701, Test Loss = 1.1462
Iteration 402: Training Loss = 1.5392, Test Loss = 1.0202
Iteration 403: Training Loss = 1.1188, Test Loss = 1.1666
Iteration 404: Training Loss = 1.5186, Test Loss = 1.0478
Iteration 405: Training Loss = 1.0500, Test Loss = 1.1601
Iteration 406: Training Loss = 1.5532, Test Loss = 1.0164
Iteration 407: Training Loss = 1.0467, Test Loss = 1.1360
Iteration 408: Training Loss = 1.5390, Test Loss = 1.0011
Iteration 409: Training Loss = 1.2020, Test Loss = 1.0886
Iteration 410: Training Loss = 1.5448, Test Loss = 1.0349
Iteration 411: Training Loss = 1.1283, Test Loss = 1.1387
Iteration 412: Training Loss = 1.5294, Test Loss = 1.0289
Iteration 413: Training Loss = 1.1186, Test Loss = 1.1922
Iteration 414: Training Loss = 1.5027, Test Loss = 1.1964
Iteration 415: Training Loss = 1.2366, Test Loss = 0.8771
Iteration 416: Training Loss = 1.7298, Test Loss = 0.8488
Iteration 417: Training Loss = 1.0668, Test Loss = 1.2544
Iteration 418: Training Loss = 1.5239, Test Loss = 1.0989
Iteration 419: Training Loss = 1.0131, Test Loss = 1.1562
Iteration 420: Training Loss = 1.5528, Test Loss = 1.0239
Iteration 421: Training Loss = 1.0910, Test Loss = 1.1627
Iteration 422: Training Loss = 1.5361, Test Loss = 1.0504
Iteration 423: Training Loss = 1.0347, Test Loss = 1.1777
Iteration 424: Training Loss = 1.5402, Test Loss = 1.0145
Iteration 425: Training Loss = 1.1280, Test Loss = 1.2005
Iteration 426: Training Loss = 1.4944, Test Loss = 1.1387
Iteration 427: Training Loss = 1.0631, Test Loss = 1.1997
Iteration 428: Training Loss = 1.5260, Test Loss = 1.0793
Iteration 429: Training Loss = 1.0906, Test Loss = 1.2157
Iteration 430: Training Loss = 1.5224, Test Loss = 1.1276
Iteration 431: Training Loss = 1.1159, Test Loss = 1.2436
Iteration 432: Training Loss = 1.5144, Test Loss = 1.1297
Iteration 433: Training Loss = 1.0784, Test Loss = 1.1986
Iteration 434: Training Loss = 1.5375, Test Loss = 1.0817
Iteration 435: Training Loss = 1.0682, Test Loss = 1.1690
Iteration 436: Training Loss = 1.5536, Test Loss = 1.0262
Iteration 437: Training Loss = 1.0534, Test Loss = 1.1432
Iteration 438: Training Loss = 1.5572, Test Loss = 1.0253
Iteration 439: Training Loss = 1.0700, Test Loss = 1.1326
Iteration 440: Training Loss = 1.5537, Test Loss = 1.0396
```

```
Iteration 441: Training Loss = 1.0791, Test Loss = 1.1567
Iteration 442: Training Loss = 1.5518, Test Loss = 1.0325
Iteration 443: Training Loss = 1.0703, Test Loss = 1.1272
Iteration 444: Training Loss = 1.5614, Test Loss = 1.0241
Iteration 445: Training Loss = 1.0720, Test Loss = 1.1503
Iteration 446: Training Loss = 1.5426, Test Loss = 1.0509
Iteration 447: Training Loss = 1.0791, Test Loss = 1.1492
Iteration 448: Training Loss = 1.5383, Test Loss = 1.0498
Iteration 449: Training Loss = 1.0333, Test Loss = 1.1448
Iteration 450: Training Loss = 1.5457, Test Loss = 1.0090
Iteration 451: Training Loss = 1.1683, Test Loss = 1.1787
Iteration 452: Training Loss = 1.5153, Test Loss = 1.1626
Iteration 453: Training Loss = 1.1001, Test Loss = 1.2535
Iteration 454: Training Loss = 1.5184, Test Loss = 1.1655
Iteration 455: Training Loss = 1.1680, Test Loss = 1.1342
Iteration 456: Training Loss = 1.5544, Test Loss = 1.0502
Iteration 457: Training Loss = 1.0236, Test Loss = 1.1202
Iteration 458: Training Loss = 1.5781, Test Loss = 0.9746
Iteration 459: Training Loss = 1.1970, Test Loss = 1.1385
Iteration 460: Training Loss = 1.5035, Test Loss = 1.1552
Iteration 461: Training Loss = 1.0568, Test Loss = 1.1741
Iteration 462: Training Loss = 1.5604, Test Loss = 1.0104
Iteration 463: Training Loss = 1.1374, Test Loss = 1.1810
Iteration 464: Training Loss = 1.5008, Test Loss = 1.1437
Iteration 465: Training Loss = 1.0861, Test Loss = 1.2366
Iteration 466: Training Loss = 1.5047, Test Loss = 1.1892
Iteration 467: Training Loss = 1.1891, Test Loss = 1.0397
Iteration 468: Training Loss = 1.6289, Test Loss = 0.9357
Iteration 469: Training Loss = 0.9416, Test Loss = 0.8451
Iteration 470: Training Loss = 1.8129, Test Loss = 0.7877
Iteration 471: Training Loss = 1.0353, Test Loss = 1.2532
Iteration 472: Training Loss = 1.5144, Test Loss = 1.0820
Iteration 473: Training Loss = 1.0325, Test Loss = 1.1232
Iteration 474: Training Loss = 1.4276, Test Loss = 1.0855
Iteration 475: Training Loss = 1.1654, Test Loss = 1.1434
Iteration 476: Training Loss = 1.5372, Test Loss = 1.0371
Iteration 477: Training Loss = 1.0155, Test Loss = 1.1305
Iteration 478: Training Loss = 1.4771, Test Loss = 1.1204
Iteration 479: Training Loss = 1.0420, Test Loss = 1.1093
Iteration 480: Training Loss = 1.5758, Test Loss = 0.9897
Iteration 481: Training Loss = 1.0111, Test Loss = 1.0477
Iteration 482: Training Loss = 1.5263, Test Loss = 1.2647
Iteration 483: Training Loss = 1.1406, Test Loss = 1.1634
Iteration 484: Training Loss = 1.5483, Test Loss = 1.0272
```

```
Iteration 485: Training Loss = 1.0338, Test Loss = 1.1263
Iteration 486: Training Loss = 1.5583, Test Loss = 0.9968
Iteration 487: Training Loss = 1.1766, Test Loss = 1.1271
Iteration 488: Training Loss = 1.5369, Test Loss = 1.0343
Iteration 489: Training Loss = 1.0975, Test Loss = 1.1489
Iteration 490: Training Loss = 1.5408, Test Loss = 1.0179
Iteration 491: Training Loss = 1.1313, Test Loss = 1.1938
Iteration 492: Training Loss = 1.4964, Test Loss = 1.1930
Iteration 493: Training Loss = 1.1853, Test Loss = 0.9706
Iteration 494: Training Loss = 1.6561, Test Loss = 0.9179
Iteration 495: Training Loss = 0.9644, Test Loss = 1.0191
Iteration 496: Training Loss = 1.6555, Test Loss = 0.8853
Iteration 497: Training Loss = 1.5317, Test Loss = 0.9696
Iteration 498: Training Loss = 1.3019, Test Loss = 0.8627
Iteration 499: Training Loss = 1.3222, Test Loss = 1.0376
Iteration 500: Training Loss = 1.5135, Test Loss = 0.7621
```


MLP Loss Curve
Training Loss
Test Loss
Loss
Iterations
0
2
4
6
8
10
12
14
0
100
200
300
400
500

```
MLP Accuracy: 87.55%
MLP Classification Report:
              precision    recall  f1-score   support

          no       0.92      0.94      0.93      1201
         yes       0.45      0.35      0.39       156

    accuracy                           0.88      1357
   macro avg       0.68      0.65      0.66      1357
weighted avg       0.86      0.88      0.87      1357

MLP Training set score: 88.56%
MLP Testing set score: 87.55%
```

In [40]:

```
##Question d Part B
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report

# Define a range of number of neurons (k)
neurons = 25  # Total number of neurons to be split between two layers
iterations_range = [100, 200, 300, 400, 500]  # Specify a range of iterations to test

# Generate combinations of neurons for two hidden layers
hidden_layer_sizes = [(k, neurons - k) for k in range(1, neurons)]

# Define parameters for the grid search
param_grid = {'hidden_layer_sizes': hidden_layer_sizes,  # Two hidden layers with varying neurons
              'max_iter': iterations_range}  # Varying number of iterations

# Create an MLPClassifier instance
mlp = MLPClassifier()

# Perform grid search with cross-validation
grid_search = GridSearchCV(mlp, param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid_search.fit(X_train, y_train)

# Get the best parameters and accuracy
best_params = grid_search.best_params_
best_accuracy = grid_search.best_score_

# Get the best iteration for the best parameters
best_iter = grid_search.cv_results_['param_max_iter'][grid_search.best_index_]
```

```
# Predict on the test set using the best parameters
y_pred = grid_search.best_estimator_.predict(X_test)

# Generate and print the classification report
#print("Classification Report:")
#print(classification_report(y_test, y_pred))

#print("Best parameters:", best_params)
#print("Best accuracy: {:.2f}%".format(best_accuracy * 100))
#print("Best iteration:", best_iter)
```

In [41]:

```
#Question d part b continued.
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
from collections import defaultdict

# Define a range of number of neurons (k)
neurons = 25  # Total number of neurons to be split between two layers
iterations_range = [100, 200, 300, 400, 500]  # Specify a range of iterations to test

# Generate combinations of neurons for two hidden layers
hidden_layer_sizes = [(k, neurons - k) for k in range(1, neurons)]

# Define parameters for the grid search
param_grid = {'hidden_layer_sizes': hidden_layer_sizes,  # Two hidden layers with varying neurons
            'max_iter': iterations_range}  # Varying number of iterations

# Create an MLPClassifier instance
mlp = MLPClassifier()

# Perform grid search with cross-validation
grid_search = GridSearchCV(mlp, param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid_search.fit(X_train, y_train)

# Get the best parameters and accuracy
best_params = grid_search.best_params_
best_accuracy = grid_search.best_score_

# Get the best iteration for the best parameters
best_iter = grid_search.cv_results_['param_max_iter'][grid_search.best_index_]

# Predict on the test set using the best parameters
```

```
y_pred = grid_search.best_estimator_.predict(X_test)

# Generate and print the classification report
#print("Classification Report:")
#print(classification_report(y_test, y_pred))

#print("Best parameters:", best_params)
#print("Best accuracy: {:.2f}%".format(best_accuracy * 100))
#print("Best iteration:", best_iter)

results = grid_search.cv_results_

# Initialize the dictionary to hold accuracies
accuracy_dict = defaultdict(list)

# Iterate over the results and populate the dictionary
for i in range(len(results['params'])):
    combination = results['params'][i]['hidden_layer_sizes']
    accuracy = results['mean_test_score'][i]
    accuracy_dict[combination].append(accuracy)

# Compute the mean accuracy for each combination
mean_accuracies = {k: sum(v) / len(v) for k, v in accuracy_dict.items()}

# Prepare the results table
results_table = [(k, v) for k, v in mean_accuracies.items()]

# Sort the results table by accuracy in descending order for better readability
results_table.sort(key=lambda x: x[1], reverse=True)

# Print the results table
print("\nResults Table in Order of Accuracy:")
print("{:<15} {:<10}".format('Combination', 'Accuracy'))
for row in results_table:
    print("{:<15} {:.2f}%".format(str(row[0]), row[1] * 100))
```

```
Results Table in Order of Accuracy:
Combination     Accuracy
(22, 3)         88.40%
(1, 24)         88.31%
(4, 21)         88.19%
(3, 22)         87.95%
(15, 10)        87.93%
(5, 20)         87.90%
(6, 19)         87.90%
(8, 17)         87.86%
(20, 5)         87.78%
(7, 18)         87.67%
(9, 16)         87.53%
(19, 6)         87.52%
(21, 4)         87.49%
(2, 23)         87.41%
(10, 15)        87.25%
(18, 7)         86.86%
(13, 12)        86.78%
(17, 8)         86.74%
(12, 13)        86.74%
(16, 9)         86.69%
(11, 14)        86.59%
(14, 11)        86.21%
(24, 1)         85.30%
(23, 2)         85.25%
```

In [42]:

```python
# Sort the results table by the first element of the combination (i.e., the number of neurons in the first hidden layer)
results_table.sort(key=lambda x: x[0][0])
# Print the results table
print("Results Table in Order of Combination:")
print("{:<15} {:<10}".format('Combination', 'Accuracy'))
for row in results_table:
    print("{:<15} {:.2f}%".format(str(row[0]), row[1] * 100))
```

```
Results Table in Order of Combination:
Combination    Accuracy
(1, 24)        88.31%
(2, 23)        87.41%
(3, 22)        87.95%
(4, 21)        88.19%
(5, 20)        87.90%
(6, 19)        87.90%
(7, 18)        87.67%
(8, 17)        87.86%
(9, 16)        87.53%
(10, 15)       87.25%
(11, 14)       86.59%
(12, 13)       86.74%
(13, 12)       86.78%
(14, 11)       86.21%
(15, 10)       87.93%
(16, 9)        86.69%
(17, 8)        86.74%
(18, 7)        86.86%
(19, 6)        87.52%
(20, 5)        87.78%
(21, 4)        87.49%
(22, 3)        88.40%
(23, 2)        85.25%
(24, 1)        85.30%
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: