In [1]:
```python
import pandas as pd
import numpy as np

from sklearn.neural_network import MLPClassifier
from sklearn.datasets import load_digits
from sklearn.model_selection import StratifiedShuffleSplit

# Load the data into a pandas DataFrame
df = pd.read_csv('hw1CIS400.csv', header=None, names=['id','diagnosis','radius_mean','texture_mean','perimeter_mean','area_mean','smoothness_mean','compactness_mean','c

# Display the first 5 rows of the DataFrame
```

In [2]:
```python
df.head()
```

Out[2]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave	...	radius_worst	texture_worst	perimeter_worst
0	data	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN
1	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	texture_worst	perimeter_worst	area_worst
2	842302	M	17.99	10.38	122.8	1001	0.1184	0.2776	0.3001	0.1471	...	17.33	184.6	2019
3	842517	M	20.57	17.77	132.9	1326	0.08474	0.07864	0.0869	0.07017	...	23.41	158.8	1956
4	84300903	M	19.69	21.25	130	1203	0.1096	0.1599	0.1974	0.1279	...	25.53	152.5	1709

5 rows × 33 columns

Out[2]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave	...	radius_worst	texture_worst	perimeter_worst
0	data	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN
1	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	texture_worst	perimeter_worst	area_worst
2	842302	M	17.99	10.38	122.8	1001	0.1184	0.2776	0.3001	0.1471	...	17.33	184.6	2019
3	842517	M	20.57	17.77	132.9	1326	0.08474	0.07864	0.0869	0.07017	...	23.41	158.8	1956
4	84300903	M	19.69	21.25	130	1203	0.1096	0.1599	0.1974	0.1279	...	25.53	152.5	1709

5 rows × 33 columns

In [3]:
```python
df.columns
```

Out[3]:
```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave', 'points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst'],
      dtype='object')
```


Out[3]:
```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave', 'points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst'],
      dtype='object')
```


In [4]:
```python
df = df.iloc[1:]
```

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

Out[5]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave	...	radius_worst	texture_worst	perimeter_worst
1	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	texture_worst	perimeter_worst	area_worst
2	842302	M	17.99	10.38	122.8	1001	0.1184	0.2776	0.3001	0.1471	...	17.33	184.6	2019
3	842517	M	20.57	17.77	132.9	1326	0.08474	0.07864	0.0869	0.07017	...	23.41	158.8	1956
4	84300903	M	19.69	21.25	130	1203	0.1096	0.1599	0.1974	0.1279	...	25.53	152.5	1709
5	84348301	M	11.42	20.38	77.58	386.1	0.1425	0.2839	0.2414	0.1052	...	26.5	98.87	567.7

5 rows × 33 columns

Out[5]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave	...	radius_worst	texture_worst	perimeter_worst
1	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	texture_worst	perimeter_worst	area_worst
2	842302	M	17.99	10.38	122.8	1001	0.1184	0.2776	0.3001	0.1471	...	17.33	184.6	2019
3	842517	M	20.57	17.77	132.9	1326	0.08474	0.07864	0.0869	0.07017	...	23.41	158.8	1956
4	84300903	M	19.69	21.25	130	1203	0.1096	0.1599	0.1974	0.1279	...	25.53	152.5	1709
5	84348301	M	11.42	20.38	77.58	386.1	0.1425	0.2839	0.2414	0.1052	...	26.5	98.87	567.7

5 rows × 33 columns

In [6]:
```python
df = df.iloc[1:]
```

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

Out[7]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave	...	radius_worst	texture_worst	perimeter_worst
2	842302	M	17.99	10.38	122.8	1001	0.1184	0.2776	0.3001	0.1471	...	17.33	184.6	2019
3	842517	M	20.57	17.77	132.9	1326	0.08474	0.07864	0.0869	0.07017	...	23.41	158.8	1956
4	84300903	M	19.69	21.25	130	1203	0.1096	0.1599	0.1974	0.1279	...	25.53	152.5	1709
5	84348301	M	11.42	20.38	77.58	386.1	0.1425	0.2839	0.2414	0.1052	...	26.5	98.87	567.7
6	84358402	M	20.29	14.34	135.1	1297	0.1003	0.1328	0.198	0.1043	...	16.67	152.2	1575
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
566	926424	M	21.56	22.39	142	1479	0.111	0.1159	0.2439	0.1389	...	26.4	166.1	2027
567	926682	M	20.13	28.25	131.2	1261	0.0978	0.1034	0.144	0.09791	...	38.25	155	1731
568	926954	M	16.6	28.08	108.3	858.1	0.08455	0.1023	0.09251	0.05302	...	34.12	126.7	1124
569	927241	M	20.6	29.33	140.1	1265	0.1178	0.277	0.3514	0.152	...	39.42	184.6	1821
570	92751	B	7.76	24.54	47.92	181	0.05263	0.04362	0	0	...	30.37	59.16	268.6

569 rows × 33 columns

Out[7]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave	...	radius_worst	texture_worst	perimeter_worst
2	842302	M	17.99	10.38	122.8	1001	0.1184	0.2776	0.3001	0.1471	...	17.33	184.6	2019
3	842517	M	20.57	17.77	132.9	1326	0.08474	0.07864	0.0869	0.07017	...	23.41	158.8	1956
4	84300903	M	19.69	21.25	130	1203	0.1096	0.1599	0.1974	0.1279	...	25.53	152.5	1709
5	84348301	M	11.42	20.38	77.58	386.1	0.1425	0.2839	0.2414	0.1052	...	26.5	98.87	567.7
6	84358402	M	20.29	14.34	135.1	1297	0.1003	0.1328	0.198	0.1043	...	16.67	152.2	1575
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
566	926424	M	21.56	22.39	142	1479	0.111	0.1159	0.2439	0.1389	...	26.4	166.1	2027
567	926682	M	20.13	28.25	131.2	1261	0.0978	0.1034	0.144	0.09791	...	38.25	155	1731
568	926954	M	16.6	28.08	108.3	858.1	0.08455	0.1023	0.09251	0.05302	...	34.12	126.7	1124
569	927241	M	20.6	29.33	140.1	1265	0.1178	0.277	0.3514	0.152	...	39.42	184.6	1821
570	92751	B	7.76	24.54	47.92	181	0.05263	0.04362	0	0	...	30.37	59.16	268.6

569 rows × 33 columns

In [8]:
```python
df.isnull().sum()
```

Out[8]:
```
id                          0
diagnosis                   0
radius_mean                 0
texture_mean                0
perimeter_mean              0
area_mean                   0
smoothness_mean             0
compactness_mean            0
concavity_mean              0
concave                     0
points_mean                 0
symmetry_mean               0
fractal_dimension_mean      0
radius_se                   0
texture_se                  0
perimeter_se                0
area_se                     0
smoothness_se               0
compactness_se              0
concavity_se                0
concave points_se           0
symmetry_se                 0
fractal_dimension_se        0
radius_worst                0
texture_worst               0
perimeter_worst             0
area_worst                  0
smoothness_worst            0
compactness_worst           0
concavity_worst             0
concave points_worst        0
symmetry_worst              0
fractal_dimension_worst   569
dtype: int64
```


Out[8]:
```
id                          0
diagnosis                   0
radius_mean                 0
texture_mean                0
perimeter_mean              0
area_mean                   0
smoothness_mean             0
compactness_mean            0
concavity_mean              0
concave                     0
points_mean                 0
symmetry_mean               0
fractal_dimension_mean      0
radius_se                   0
texture_se                  0
perimeter_se                0
area_se                     0
smoothness_se               0
compactness_se              0
concavity_se                0
concave points_se           0
symmetry_se                 0
fractal_dimension_se        0
radius_worst                0
texture_worst               0
perimeter_worst             0
area_worst                  0
smoothness_worst            0
compactness_worst           0
concavity_worst             0
concave points_worst        0
symmetry_worst              0
fractal_dimension_worst   569
dtype: int64
```


In [9]:
```python
df = df.drop(columns=['fractal_dimension_worst'])
```

In [10]:
```python
df
```

Out[10]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave	...	fractal_dimension_se	radius_worst	texture_wo
2	842302	M	17.99	10.38	122.8	1001	0.1184	0.2776	0.3001	0.1471	...	25.38	17.33	18
3	842517	M	20.57	17.77	132.9	1326	0.08474	0.07864	0.0869	0.07017	...	24.99	23.41	15
4	84300903	M	19.69	21.25	130	1203	0.1096	0.1599	0.1974	0.1279	...	23.57	25.53	15
5	84348301	M	11.42	20.38	77.58	386.1	0.1425	0.2839	0.2414	0.1052	...	14.91	26.5	98
6	84358402	M	20.29	14.34	135.1	1297	0.1003	0.1328	0.198	0.1043	...	22.54	16.67	15
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
566	926424	M	21.56	22.39	142	1479	0.111	0.1159	0.2439	0.1389	...	25.45	26.4	16
567	926682	M	20.13	28.25	131.2	1261	0.0978	0.1034	0.144	0.09791	...	23.69	38.25	1
568	926954	M	16.6	28.08	108.3	858.1	0.08455	0.1023	0.09251	0.05302	...	18.98	34.12	12
569	927241	M	20.6	29.33	140.1	1265	0.1178	0.277	0.3514	0.152	...	25.74	39.42	18
570	92751	B	7.76	24.54	47.92	181	0.05263	0.04362	0	0	...	9.456	30.37	59

569 rows × 32 columns

Out[10]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave	...	fractal_dimension_se	radius_worst	texture_wo
2	842302	M	17.99	10.38	122.8	1001	0.1184	0.2776	0.3001	0.1471	...	25.38	17.33	18
3	842517	M	20.57	17.77	132.9	1326	0.08474	0.07864	0.0869	0.07017	...	24.99	23.41	15
4	84300903	M	19.69	21.25	130	1203	0.1096	0.1599	0.1974	0.1279	...	23.57	25.53	15
5	84348301	M	11.42	20.38	77.58	386.1	0.1425	0.2839	0.2414	0.1052	...	14.91	26.5	98
6	84358402	M	20.29	14.34	135.1	1297	0.1003	0.1328	0.198	0.1043	...	22.54	16.67	15
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
566	926424	M	21.56	22.39	142	1479	0.111	0.1159	0.2439	0.1389	...	25.45	26.4	16
567	926682	M	20.13	28.25	131.2	1261	0.0978	0.1034	0.144	0.09791	...	23.69	38.25	1
568	926954	M	16.6	28.08	108.3	858.1	0.08455	0.1023	0.09251	0.05302	...	18.98	34.12	12
569	927241	M	20.6	29.33	140.1	1265	0.1178	0.277	0.3514	0.152	...	25.74	39.42	18
570	92751	B	7.76	24.54	47.92	181	0.05263	0.04362	0	0	...	9.456	30.37	59

569 rows × 32 columns

In [11]:
```python
df.isnull().sum()
```

Out[11]:
```
id                        0
diagnosis                 0
radius_mean               0
texture_mean              0
perimeter_mean            0
area_mean                 0
smoothness_mean           0
compactness_mean          0
concavity_mean            0
concave                   0
points_mean               0
symmetry_mean             0
fractal_dimension_mean    0
radius_se                 0
texture_se                0
perimeter_se              0
area_se                   0
smoothness_se             0
compactness_se            0
concavity_se              0
concave points_se         0
symmetry_se               0
fractal_dimension_se      0
radius_worst              0
texture_worst             0
perimeter_worst           0
area_worst                0
smoothness_worst          0
compactness_worst         0
concavity_worst           0
concave points_worst      0
symmetry_worst            0
dtype: int64
```


Out[11]:
```
id                        0
diagnosis                 0
radius_mean               0
texture_mean              0
perimeter_mean            0
area_mean                 0
smoothness_mean           0
compactness_mean          0
concavity_mean            0
concave                   0
points_mean               0
symmetry_mean             0
fractal_dimension_mean    0
radius_se                 0
texture_se                0
perimeter_se              0
area_se                   0
smoothness_se             0
compactness_se            0
concavity_se              0
concave points_se         0
symmetry_se               0
fractal_dimension_se      0
radius_worst              0
texture_worst             0
perimeter_worst           0
area_worst                0
smoothness_worst          0
compactness_worst         0
concavity_worst           0
concave points_worst      0
symmetry_worst            0
dtype: int64
```


In [12]:
```python
# Determine the class distribution
class_distribution = df['diagnosis'].value_counts()

# Determine the minority class
minority_class = class_distribution.idxmin()

# Upsample the minority class
df_minority_upsampled = df[df['diagnosis'] == minority_class].sample(n=class_distribution[minority_class], replace=True)

# Concatenate the upsampled minority class with the original data frame
df_balanced = pd.concat([df, df_minority_upsampled])

# Shuffle the rows of the balanced data frame
df_balanced = df_balanced.sample(frac=1)

# Split the data frame into training and testing sets
training_set = df_balanced[:int(0.7 * df_balanced.shape[0])]
testing_set = df_balanced[int(0.7 * df_balanced.shape[0]):]
```

In [13]:
```python
# Check the class distribution
class_counts = df['diagnosis'].value_counts()

# Print the class distribution
print(class_counts)
```

```
B    357
M    212
Name: diagnosis, dtype: int64
B    357
M    212
Name: diagnosis, dtype: int64
```

In [16]:
```python
# Load the data
digits = load_digits()
x = digits.images.reshape(digits.images.shape[0], -1)
y = digits.target
```

In [17]:
```python
# Split the data into training and test sets, ensuring balance
sss = StratifiedShuffleSplit(n_splits=1, test_size=0.3, random_state=0)
train_index, test_index = next(sss.split(x, y))
x_train, x_test = x[train_index], x[test_index]
y_train, y_test = y[train_index], y[test_index]
```

In [18]:
```python
# Train the model
mlp = MLPClassifier(hidden_layer_sizes=(x_train.shape[1] * 2,), activation='logistic', solver='adam', max_iter=500)
mlp.fit(x_train, y_train)

# Evaluate the model
accuracy = mlp.score(x_test, y_test)
print('Test accuracy:', accuracy)
```

```
Test accuracy: 0.9814814814814815
Test accuracy: 0.9814814814814815
```

In [19]:
```python
# Load the data
digits = load_digits()
x = digits.images.reshape(digits.images.shape[0], -1)
y = digits.target
```

In [20]:
```python
# Split the data into training and test sets, ensuring balance
sss1 = StratifiedShuffleSplit(n_splits=1, test_size=0.3, random_state=10)
train_index, test_index = next(sss.split(x, y))
x_train, x_test = x[train_index], x[test_index]
y_train, y_test = y[train_index], y[test_index]
```

In [21]:
```python
# Train the model
mlp = MLPClassifier(hidden_layer_sizes=(x_train.shape[1] * 2,), activation='logistic', solver='adam', max_iter=500)
mlp.fit(x_train, y_train)

# Evaluate the model
accuracy = mlp.score(x_test, y_test)
print('Test accuracy:', accuracy)
```

```
Test accuracy: 0.9796296296296296
Test accuracy: 0.9796296296296296
```

In [22]:
```python
# Load the data
digits = load_digits()
x = digits.images.reshape(digits.images.shape[0], -1)
y = digits.target
```

In [23]:
```python
# Split the data into training and test sets, ensuring balance
sss2 = StratifiedShuffleSplit(n_splits=1, test_size=0.3, random_state=20)
train_index, test_index = next(sss.split(x, y))
x_train, x_test = x[train_index], x[test_index]
y_train, y_test = y[train_index], y[test_index]
```

In [24]:
```python
# Train the model
mlp = MLPClassifier(hidden_layer_sizes=(x_train.shape[1] * 2,), activation='logistic', solver='adam', max_iter=500)
mlp.fit(x_train, y_train)

# Evaluate the model
accuracy = mlp.score(x_test, y_test)
print('Test accuracy:', accuracy)
```

```
Test accuracy: 0.9777777777777777
Test accuracy: 0.9777777777777777
```

In [25]:
```python
# Load the data
digits = load_digits()
x = digits.images.reshape(digits.images.shape[0], -1)
y = digits.target
```

In [26]:
```python
# Split the data into training and test sets, ensuring balance
sss3 = StratifiedShuffleSplit(n_splits=1, test_size=0.3, random_state=30)
train_index, test_index = next(sss.split(x, y))
x_train, x_test = x[train_index], x[test_index]
y_train, y_test = y[train_index], y[test_index]
```

In [27]:
```python
# Train the model
mlp = MLPClassifier(hidden_layer_sizes=(x_train.shape[1] * 2,), activation='logistic', solver='adam', max_iter=500)
mlp.fit(x_train, y_train)

# Evaluate the model
accuracy = mlp.score(x_test, y_test)
print('Test accuracy:', accuracy)
```

```
Test accuracy: 0.9796296296296296
Test accuracy: 0.9796296296296296
```

In [28]:
```python
# Load the data
digits = load_digits()
x = digits.images.reshape(digits.images.shape[0], -1)
y = digits.target
```

In [29]:
```python
# Split the data into training and test sets, ensuring balance
sss4 = StratifiedShuffleSplit(n_splits=1, test_size=0.3, random_state=40)
train_index, test_index = next(sss.split(x, y))
x_train, x_test = x[train_index], x[test_index]
y_train, y_test = y[train_index], y[test_index]
```

In [30]:
```python
# Train the model
mlp = MLPClassifier(hidden_layer_sizes=(x_train.shape[1] * 2,), activation='logistic', solver='adam', max_iter=500)
mlp.fit(x_train, y_train)

# Evaluate the model
accuracy = mlp.score(x_test, y_test)
print('Test accuracy:', accuracy)
```

```
Test accuracy: 0.9814814814814815
Test accuracy: 0.9814814814814815
```

In [31]:
```python
# Load the data
digits = load_digits()
x = digits.images.reshape(digits.images.shape[0], -1)
y = digits.target
```

In [32]:
```python
# Split the data into training and test sets, ensuring balance
sss5 = StratifiedShuffleSplit(n_splits=1, test_size=0.3, random_state=55)
train_index, test_index = next(sss.split(x, y))
x_train, x_test = x[train_index], x[test_index]
y_train, y_test = y[train_index], y[test_index]
```

In [33]:
```python
# Train the model
mlp = MLPClassifier(hidden_layer_sizes=(x_train.shape[1] * 2,), activation='logistic', solver='adam', max_iter=500)
mlp.fit(x_train, y_train)

# Evaluate the model
accuracy = mlp.score(x_test, y_test)
print('Test accuracy:', accuracy)
```

```
Test accuracy: 0.9814814814814815
Test accuracy: 0.9814814814814815
```

In [34]:
```python
# Load the data
digits = load_digits()
x = digits.images.reshape(digits.images.shape[0], -1)
y = digits.target
```

In [37]:
```python
# Split the data into training and test sets, ensuring balance
sss6 = StratifiedShuffleSplit(n_splits=1, test_size=0.3, random_state=63)
train_index, test_index = next(sss.split(x, y))
x_train, x_test = x[train_index], x[test_index]
y_train, y_test = y[train_index], y[test_index]
```

In [38]:
```python
# Train the model
mlp = MLPClassifier(hidden_layer_sizes=(x_train.shape[1] * 2,), activation='logistic', solver='adam', max_iter=500)
mlp.fit(x_train, y_train)

# Evaluate the model
accuracy = mlp.score(x_test, y_test)
print('Test accuracy:', accuracy)
```

```
Test accuracy: 0.9814814814814815
Test accuracy: 0.9814814814814815
```

In [39]:
```python
# Load the data
digits = load_digits()
x = digits.images.reshape(digits.images.shape[0], -1)
y = digits.target
```

In [40]:
```python
# Split the data into training and test sets, ensuring balance
sss7 = StratifiedShuffleSplit(n_splits=1, test_size=0.3, random_state=78)
train_index, test_index = next(sss.split(x, y))
x_train, x_test = x[train_index], x[test_index]
y_train, y_test = y[train_index], y[test_index]
```

In [41]:
```python
# Train the model
mlp = MLPClassifier(hidden_layer_sizes=(x_train.shape[1] * 2,), activation='logistic', solver='adam', max_iter=500)
mlp.fit(x_train, y_train)

# Evaluate the model
accuracy = mlp.score(x_test, y_test)
print('Test accuracy:', accuracy)
```

```
Test accuracy: 0.9777777777777777
Test accuracy: 0.9777777777777777
```

In [42]:
```python
# Load the data
digits = load_digits()
x = digits.images.reshape(digits.images.shape[0], -1)
y = digits.target
```

In [43]:
```python
# Split the data into training and test sets, ensuring balance
sss8 = StratifiedShuffleSplit(n_splits=1, test_size=0.3, random_state=13)
train_index, test_index = next(sss.split(x, y))
x_train, x_test = x[train_index], x[test_index]
y_train, y_test = y[train_index], y[test_index]
```

In [44]:
```python
# Train the model
mlp = MLPClassifier(hidden_layer_sizes=(x_train.shape[1] * 2,), activation='logistic', solver='adam', max_iter=500)
mlp.fit(x_train, y_train)

# Evaluate the model
accuracy = mlp.score(x_test, y_test)
print('Test accuracy:', accuracy)
```

```
Test accuracy: 0.9851851851851852
Test accuracy: 0.9851851851851852
```

In [45]:
```python
# Load the data
digits = load_digits()
x = digits.images.reshape(digits.images.shape[0], -1)
y = digits.target
```

In [46]:
```python
# Split the data into training and test sets, ensuring balance
sss9 = StratifiedShuffleSplit(n_splits=1, test_size=0.3, random_state=33)
train_index, test_index = next(sss.split(x, y))
x_train, x_test = x[train_index], x[test_index]
y_train, y_test = y[train_index], y[test_index]
```

In [47]:
```python
# Train the model
mlp = MLPClassifier(hidden_layer_sizes=(x_train.shape[1] * 2,), activation='logistic', solver='adam', max_iter=500)
mlp.fit(x_train, y_train)

# Evaluate the model
accuracy = mlp.score(x_test, y_test)
print('Test accuracy:', accuracy)
```

```
Test accuracy: 0.9796296296296296
Test accuracy: 0.9796296296296296
```

In [48]:
```python
# Load the data
digits = load_digits()
x = digits.images.reshape(digits.images.shape[0], -1)
y = digits.target
```

In [51]:
```python
# Split the data into training and test sets, ensuring balance
sss10 = StratifiedShuffleSplit(n_splits=1, test_size=0.3, random_state=102)
train_index, test_index = next(sss.split(x, y))
x_train, x_test = x[train_index], x[test_index]
y_train, y_test = y[train_index], y[test_index]
```

In [52]:
```python
# Train the model
mlp = MLPClassifier(hidden_layer_sizes=(x_train.shape[1] * 2,), activation='logistic', solver='adam', max_iter=500)
mlp.fit(x_train, y_train)

# Evaluate the model
accuracy = mlp.score(x_test, y_test)
print('Test accuracy:', accuracy)
```

```
Test accuracy: 0.9814814814814815
Test accuracy: 0.9814814814814815
```

In [ ]: