นาย สัณหณัฐ พรมจรรย์ 63070501069

Quiz 8

1. 15 points. 1 hour. Using the Wine dataset, use PCA to reduce it to 2 PCA features.

Link: https://colab.research.google.com/drive/1r8lydpJCCCoyEPipXjkAmALvcCK3y5k-?usp=sharing

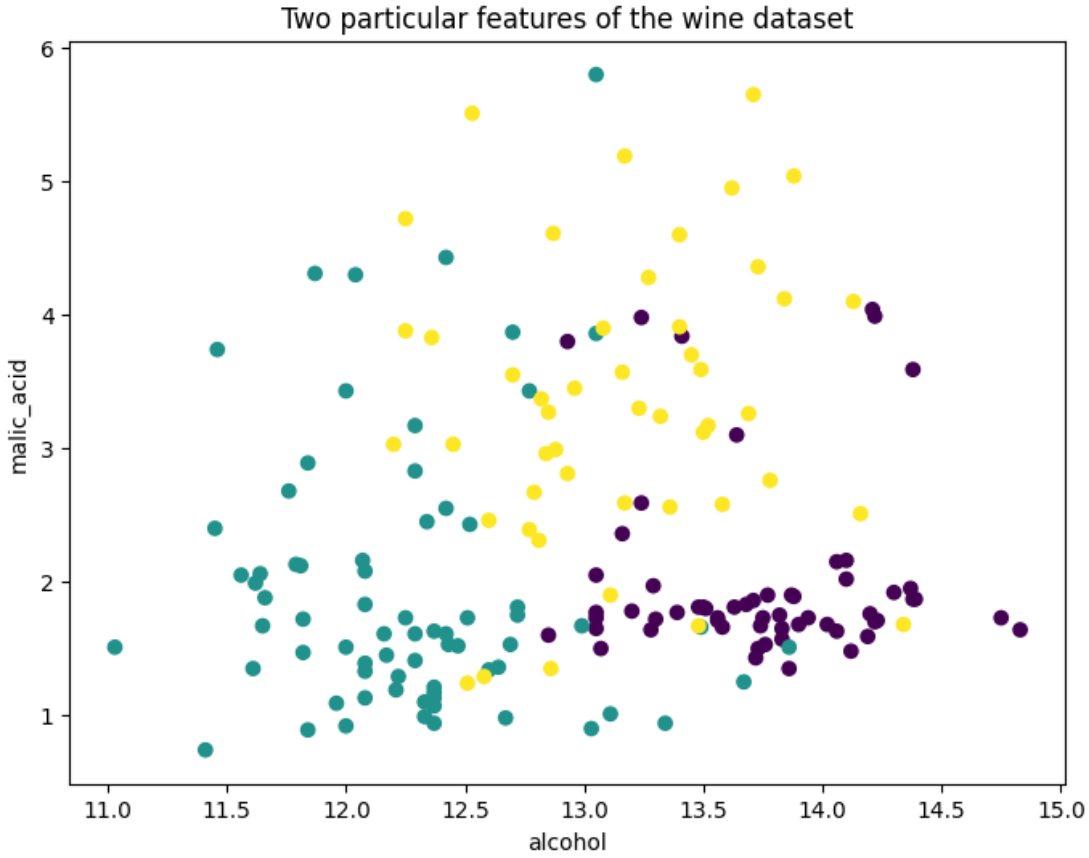
[1]
```
#copied mainly from https://machinelearningmastery.com/principal-component-analysis-for-visualization/
from sklearn.datasets import load_wine
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
import matplotlib.pyplot as plt
import numpy as np
```

[2]
```
# Load dataset
winedata = load_wine()
X = winedata['data']
y = winedata['target']
print("X shape:", X.shape)
print("y shape:", y.shape)
print("Target Output: \n", y) #y has 3 classes
print("Feature Names: \n", winedata["feature_names"])
print("Target Names: \n", winedata["target_names"])

```

```
X shape: (178, 13)
y shape: (178,)
Target Output:
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
Feature Names:
 ['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium', 'total_phenols', 'flavanoids', 'nonflavanoid_phenols', 'pro
Target Names:
 ['class_0' 'class_1' 'class_2']
```

1.1. Visualize the first 2 columns as a scatter plot using all (100%) of the data.

```
[3]  1 # Scatter plot any two features f1 and f2
     2 f1 = 0
     3 f2 = 1
     4 plt.figure(figsize=(8,6))
     5 plt.scatter(X[:,f1], X[:,f2], c=y)
     6 plt.xlabel(winedata["feature_names"][f1])
     7 plt.ylabel(winedata["feature_names"][f2])
     8 plt.title("Two particular features of the wine dataset")
     9 plt.show()
```



1.2. In another graph visualize the scatter plot for the 2 PCA features also using all the data.

[4]
```
# Show first two principal components with scaler
pca = PCA(n_components=2)
pipe = Pipeline([('scaler', StandardScaler()), ('pca', pca)])
plt.figure(figsize=(8,6))
Xt = pipe.fit_transform(X)
plot = plt.scatter(Xt[:,0], Xt[:,1], c=y)
plt.legend(handles=plot.legend_elements()[0], labels=list(winedata['target_names']))
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.title("First two principal components after scaling training data")
plt.show()
```

1.3. What % of information ("explained variance") is preserved in the 2 PCA features?

[5]
```
# Print the explained variance ratio
print("Explained variance ratios:")
print(pca.explained_variance_ratio_)
```

```
Explained variance ratios:
[0.36198848 0.1920749 ]
```

[6]
```
print("Explained variance of the PCA features: {:.2f}%".format(np.sum(pca.explained_variance_ratio_) * 100))
```

```
Explained variance of the PCA features: 55.41%
```

2. 10 points. 1 hour. Use 25% for testing for the Wine dataset. Plot a graph of the accuracy of the samples when using 1, 2, 3, …, 13 PCA features along with showing how much % variance is preserved (p%) for each? Use logistic regression.

Link: https://colab.research.google.com/drive/1PlcOjJH6U9aHv_3h9MZE7-ho5EExaXhf?usp=sharing

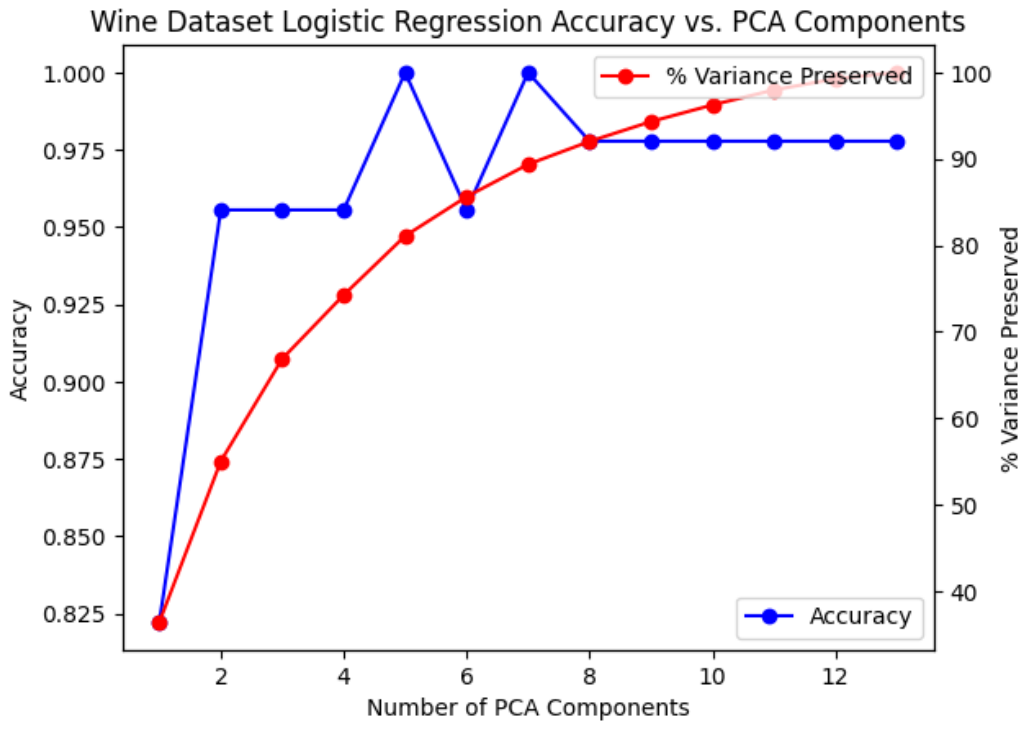
[1]
```
#copied mainly from https://machinelearningmastery.com/principal-component-analysis-for-visualization/
from sklearn.datasets import load_wine
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
import matplotlib.pyplot as plt
import numpy as np
```

[2]
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

[3]
```
# Load the Wine dataset
wine = load_wine()

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(wine.data, wine.target, test_size=0.25, random_state=42)

# Standardize the features using StandardScaler
scaler = StandardScaler()
X_train_std = scaler.fit_transform(X_train)
X_test_std = scaler.transform(X_test)

# Compute the principal components
pca = PCA()
pca.fit(X_train_std)

# Compute the variance explained by each component
var_exp = pca.explained_variance_ratio_

# Compute the cumulative variance explained
cum_var_exp = np.cumsum(var_exp)

# Initialize lists to store the accuracies and % variance preserved
accuracies = []
percent_var_preserved = []
```

[4]

```python
# Loop over the number of PCA components
for i in range(1, 14):
    # Compute the first i principal components
    pca = PCA(n_components=i)
    pca.fit(X_train_std)
    X_train_pca = pca.transform(X_train_std)
    X_test_pca = pca.transform(X_test_std)

    # Fit a logistic regression model to the transformed data
    lr = LogisticRegression(random_state=42)
    lr.fit(X_train_pca, y_train)

    # Compute the accuracy on the transformed testing data
    accuracy = lr.score(X_test_pca, y_test)
    accuracies.append(accuracy)

    # Compute the % variance preserved
    percent_var = cum_var_exp[i-1]*100
    percent_var_preserved.append(percent_var)

# Plot the accuracy vs. the number of PCA components
plt.plot(range(1, 14), accuracies, 'bo-', label='Accuracy')
plt.xlabel('Number of PCA Components')
plt.ylabel('Accuracy')
plt.title('Wine Dataset Logistic Regression Accuracy vs. PCA Components')
plt.legend(loc='lower right')

# Plot the % variance preserved vs. the number of PCA components
plt.twinx()
plt.plot(range(1, 14), percent_var_preserved, 'ro-', label='% Variance Preserved')
plt.ylabel('% Variance Preserved')
plt.legend(loc='upper right')

plt.show()
```



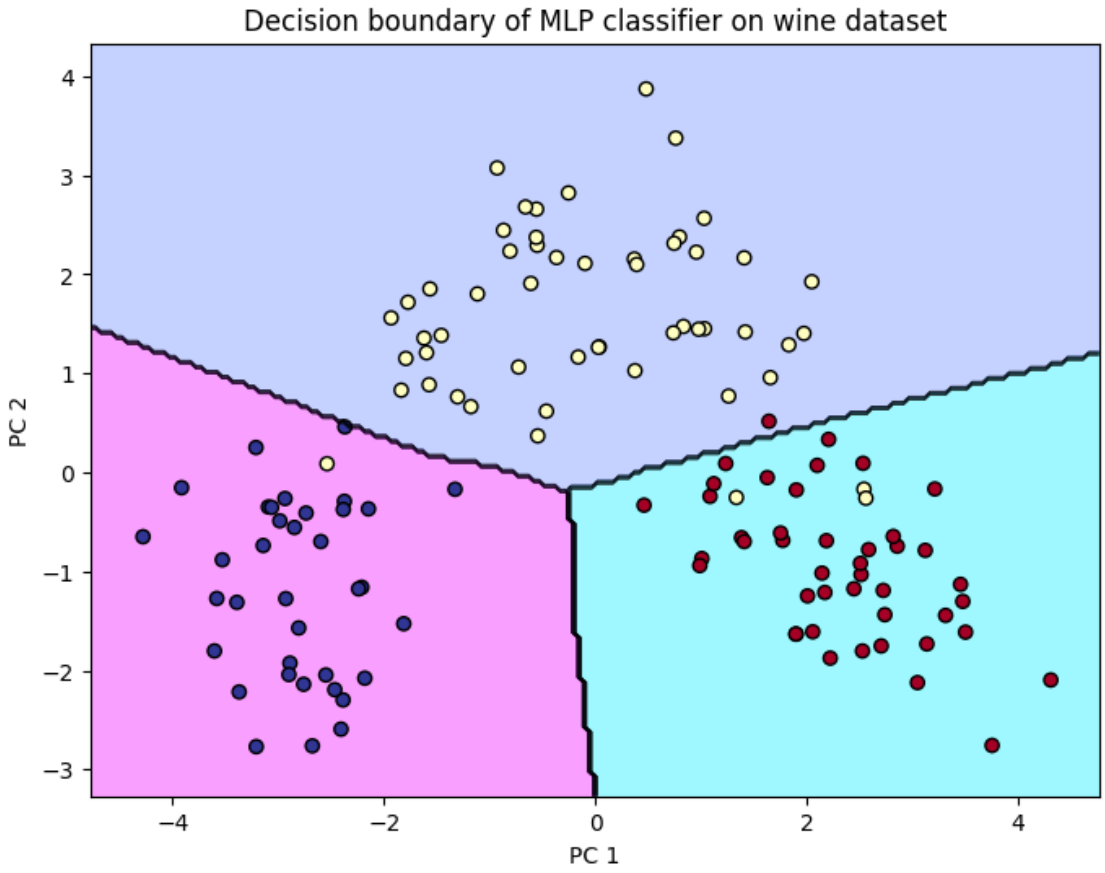
3. 15 points. 2 hours. Decision boundary. Plot the decision boundary for PCA-reduced 2 dimensions of the wine dataset. Use ML Perceptron in https://python-course.eu/machine-learning/neural-networks-with-scikit.php with 2 hidden layers.

Link: https://colab.research.google.com/drive/1tmF-MHHgtq0zERJgn0RysYCx30RaVEEN?usp=sharing

[1]
```
# Load necessary libraries and modules
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
```

[2]
```
# Load the wine dataset
wine_data = load_wine()
X = wine_data['data']
y = wine_data['target']

# Perform PCA on the training data
pca = PCA(n_components=2)
scaler = StandardScaler()
X_pca = pca.fit_transform(scaler.fit_transform(X))

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_pca, y, test_size=0.25, random_state=42)

# Train a MLP classifier with 2 hidden layers
model = MLPClassifier(hidden_layer_sizes=(10, 10), max_iter=1000)
model.fit(X_train, y_train)

# Plot the decision boundary
h = 0.05
x_min, x_max = X_train[:, 0].min() - 0.5, X_train[:, 0].max() + 0.5
y_min, y_max = X_train[:, 1].min() - 0.5, X_train[:, 1].max() + 0.5
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
```

[3]
```
# Create the plot
plt.figure(figsize=(8, 6))
plt.contourf(xx, yy, Z, cmap=plt.cm.cool, alpha=0.4)
plt.contour(xx, yy, Z, colors='black', linewidths=0.5)
plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=plt.cm.RdYlBu, edgecolors='black')
plt.xlabel('PC 1')
plt.ylabel('PC 2')
plt.title('Decision boundary of MLP classifier on wine dataset')
plt.show()
```



4. Using the MNIST data of 784 dimensions (28 x 28 pixels).

Link: https://colab.research.google.com/drive/1wVTUEd6ipaYL4iOpcUoVsBDmXxUPOkHJ?usp=sharing

4.1. 5 points. 0.5 hours. Show Quiz 7 problem #3 here again as a benchmark. Using 50,000 data samples for training and 20,000 for testing, apply standard scalar transform from the training dataset to both your training and testing datasets. Report the accuracy and print out the confusion matrix.

[25]
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_openml
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn import metrics
from sklearn.preprocessing import StandardScaler
```

[26]
```
# Load the MNIST dataset
mnist = fetch_openml("mnist_784")

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(mnist.data, mnist.target, train_size=50000, test_size=20000)

# Scale the data to be between 0 and 1
X_train = X_train / 255.0
X_test = X_test / 255.0
```

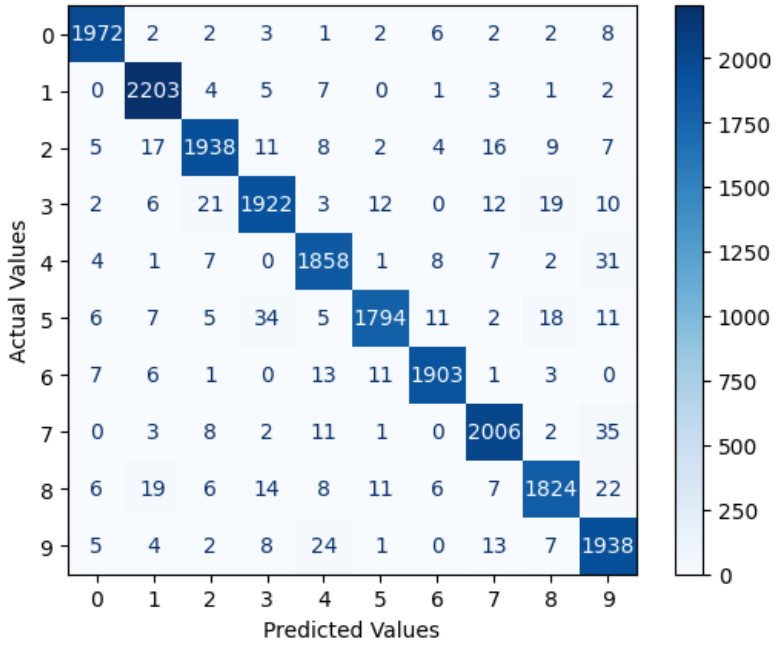
```
/usr/local/lib/python3.9/dist-packages/sklearn/datasets/_openml.py:968: FutureWarning: The default value of `parser` will c
  warn(
```

[27]
```
# Initialize the MLPClassifier
clf = MLPClassifier(hidden_layer_sizes=(100, 100), max_iter=10)

# Train the MLPClassifier on the training data
clf.fit(X_train, y_train)

# Use the trained MLPClassifier to make predictions on the testing data
y_pred = clf.predict(X_test)

# Scale the training set
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)

# Scale the testing set using the same scaler
X_test_scaled = scaler.transform(X_test)
```

[28]
```
# Initialize the MLPClassifier
clf = MLPClassifier(hidden_layer_sizes=(100, 100), max_iter=10)

# Train the MLPClassifier on the training data
clf.fit(X_train_scaled, y_train)

# Use the trained MLPClassifier to make predictions on the testing data
y_pred = clf.predict(X_test_scaled)

# Print the accuracy score of the MLPClassifier on the testing data
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy:', accuracy)

# Show the confusion matrix of the MLPClassifier on the testing data
print('\nConfusion Matrix:')
confusion_matrix(y_test, y_pred)
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stocha
  warnings.warn(
Accuracy: 0.9679

Confusion Matrix:
array([[1972,    2,    2,    3,    1,    2,    6,    2,    2,    8],
       [   0, 2203,    4,    5,    7,    0,    1,    3,    1,    2],
       [   5,   17, 1938,   11,    8,    2,    4,   16,    9,    7],
       [   2,    6,   21, 1922,    3,   12,    0,   12,   19,   10],
       [   4,    1,    7,    0, 1858,    1,    8,    7,    2,   31],
       [   6,    7,    5,   34,    5, 1794,   11,    2,   18,   11],
       [   7,    6,    1,    0,   13,   11, 1903,    1,    3,    0],
       [   0,    3,    8,    2,   11,    1,    0, 2006,    2,   35],
       [   6,   19,    6,   14,    8,   11,    6,    7, 1824,   22],
       [   5,    4,    2,    8,   24,    1,    0,   13,    7, 1938]])
```

[29]
```
# Confusion matrix
cm = metrics.confusion_matrix(y_test, y_pred)
classes = np.unique(mnist.target)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=classes)
cm_display.plot(cmap=plt.cm.Blues)
plt.xlabel('Predicted Values')
plt.ylabel('Actual Values')
plt.show()
```



4.2. 20 points. 2 hours. Using the same neural network model as in 4.1 above, but using the PCA features that preserve up to 95% of the information content, find a new training and testing dataset using PCA transformation based on the training dataset. Apply standard scalar transform from the training dataset to both your training and testing datasets. Report the accuracy and print out the confusion matrix for PCA with reduced features.

[41]
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_openml
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn import metrics
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline
```

[42]
```
# Load the MNIST dataset
X, y = fetch_openml("mnist_784", version=1, return_X_y=True)

# Normalize intensity of images to make it in the range [0,1] since 255 is the max (white).
X = X / 255.0

# Split the data into train/test sets
X_train, X_test = X[:50000], X[20000:]
y_train, y_test = y[:50000], y[20000:]
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/datasets/_openml.py:968: FutureWarning: The default value of `parser` will change from
  warn(
```

[43]
```
# Apply PCA to training set with 95% variance preserved
pca = PCA(n_components=0.95)
pipe = Pipeline([('scaler', StandardScaler()), ('pca', pca)])
X_train_pca = pipe.fit_transform(X_train)
X_test_pca = pipe.transform(X_test)
X_train_pca = pca.fit_transform(X_train)

# Apply PCA to testing set with the same transformation
X_test_pca = pca.transform(X_test)

# Train neural network model
classifier = MLPClassifier(hidden_layer_sizes=(50,), max_iter=10, alpha=1e-4,
                           solver='sgd', verbose=10, random_state=1,
                           learning_rate_init=.1)

clf = classifier.fit(X_train_pca, y_train)

# Make predictions on test data
y_pred = clf.predict(X_test_pca)
```

```
Iteration 1, loss = 0.32053499
Iteration 2, loss = 0.14439844
Iteration 3, loss = 0.11033504
Iteration 4, loss = 0.09065064
Iteration 5, loss = 0.07793818
Iteration 6, loss = 0.06816407
Iteration 7, loss = 0.05963315
Iteration 8, loss = 0.05319021
Iteration 9, loss = 0.04850037
Iteration 10, loss = 0.04360186
/usr/local/lib/python3.9/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer:
  warnings.warn(
```

[44]
```
# Print the accuracy score of the MLPClassifier on the testing data
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy:', accuracy)

# Show the confusion matrix of the MLPClassifier on the testing data
print('\nConfusion Matrix:')
confusion_matrix(y_test, y_pred)
```

```
Accuracy: 0.9833

Confusion Matrix:
array([[4879,    1,   10,    2,    2,    1,    4,    3,    2,    5],
       [   0, 5565,    6,    4,    2,    1,    7,    8,    3,    0],
       [  11,   13, 4969,    8,   13,    0,    4,   27,   14,    2],
       [   4,    2,   25, 4983,    0,   16,    0,    7,   15,   13],
       [   4,   13,    2,    0, 4789,    0,   11,    6,    2,   52],
       [  13,    5,    3,   51,    5, 4411,   24,    3,    9,   14],
       [  15,    4,    3,    2,   10,    6, 4861,    1,    3,    0],
       [   4,   19,   15,    8,    7,    3,    0, 5118,    2,   24],
       [  15,   16,   16,   42,    2,   13,   14,    7, 4760,   18],
       [   9,    8,    2,   21,   24,   16,    0,   30,    4, 4830]])
```

[45]
```
# Confusion matrix
cm = metrics.confusion_matrix(y_test, y_pred)
classes = np.unique(mnist.target)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=classes)
cm_display.plot(cmap=plt.cm.Blues)
plt.xlabel('Predicted Values')
plt.ylabel('Actual Values')
plt.show()
```