10_exercise_pca_robertson

April 20, 2025

Exercise 10: Principal Component Analysis

CPSC 381/581: Machine Learning

Yale University

Instructor: Alex Wong

Student: Hailey Robertson

Prerequisites:

1. Enable Google Colaboratory as an app on your Google Drive account
2. Create a new Google Colab notebook, this will also create a "Colab Notebooks" directory under "MyDrive" i.e.

```
/content/drive/MyDrive/Colab Notebooks
```

3. Create the following directory structure in your Google Drive

```
/content/drive/MyDrive/Colab Notebooks/CPSC 381-581: Machine Learning/Exercises
```

4. Move the 10_exercise_pca.ipynb into

```
/content/drive/MyDrive/Colab Notebooks/CPSC 381-581: Machine Learning/Exercises
```

so that its absolute path is

```
/content/drive/MyDrive/Colab Notebooks/CPSC 381-581: Machine Learning/Exercises/10_exercise_pca
```

In this exercise, we will using PCA for dimensionality reduction as a mean of visualizing high dimensional data. Then we will test out the loss as we decrease the number of principal components. Finally, we will use it as a feature extractor and show that we can compress the data for the downstream classification task.

Submission:

1. Implement all TODOs in the code blocks below.
2. Report your reconstruction loss for training and testing sets and your classification scores for training and validation sets.

```
***** Fitting PCA with 1 components on iris dataset *****
Training set mean squared error: 0.0826
Validation set mean squared error: 0.1002
***** Fitting PCA with 2 components on iris dataset *****
```

```
Training set mean squared error: 0.0256
Validation set mean squared error: 0.0251
***** Fitting PCA with 3 components on iris dataset *****
Training set mean squared error: 0.0060
Validation set mean squared error: 0.0062
***** Fitting PCA with 4 components on iris dataset *****
Training set mean squared error: 0.0000
Validation set mean squared error: 0.0000

***** Fitting PCA with 1 components on wine dataset *****
Training set mean squared error: 14.4902
Validation set mean squared error: 14.6765
***** Fitting PCA with 2 components on wine dataset *****
Training set mean squared error: 1.3273
Validation set mean squared error: 1.3065
***** Fitting PCA with 3 components on wine dataset *****
Training set mean squared error: 0.6279
Validation set mean squared error: 0.4718
***** Fitting PCA with 4 components on wine dataset *****
Training set mean squared error: 0.2152
Validation set mean squared error: 0.2000
***** Fitting PCA with 5 components on wine dataset *****
Training set mean squared error: 0.1211
Validation set mean squared error: 0.1037
***** Fitting PCA with 6 components on wine dataset *****
Training set mean squared error: 0.0538
Validation set mean squared error: 0.0486
***** Fitting PCA with 7 components on wine dataset *****
Training set mean squared error: 0.0310
Validation set mean squared error: 0.0312
***** Fitting PCA with 8 components on wine dataset *****
Training set mean squared error: 0.0191
Validation set mean squared error: 0.0216
***** Fitting PCA with 9 components on wine dataset *****
Training set mean squared error: 0.0102
Validation set mean squared error: 0.0131
***** Fitting PCA with 10 components on wine dataset *****
Training set mean squared error: 0.0047
Validation set mean squared error: 0.0073
***** Fitting PCA with 11 components on wine dataset *****
Training set mean squared error: 0.0022
Validation set mean squared error: 0.0025
***** Fitting PCA with 12 components on wine dataset *****
Training set mean squared error: 0.0006
Validation set mean squared error: 0.0006
***** Fitting PCA with 13 components on wine dataset *****
Training set mean squared error: 0.0000
Validation set mean squared error: 0.0000
```

```
***** Results of Logistic Regression using PCA with 1 components on iris dataset *****
Training set mean accuracy: 0.9417
Validation set mean accuracy: 0.9333
***** Results of Logistic Regression using PCA with 2 components on iris dataset *****
Training set mean accuracy: 0.9667
Validation set mean accuracy: 0.9667
***** Results of Logistic Regression using PCA with 3 components on iris dataset *****
Training set mean accuracy: 0.9750
Validation set mean accuracy: 1.0000
***** Results of Logistic Regression using PCA with 4 components on iris dataset *****
Training set mean accuracy: 0.9750
Validation set mean accuracy: 1.0000

***** Results of Logistic Regression using PCA with 1 components on wine dataset *****
Training set mean accuracy: 0.6972
Validation set mean accuracy: 0.6667
***** Results of Logistic Regression using PCA with 2 components on wine dataset *****
Training set mean accuracy: 0.7042
Validation set mean accuracy: 0.6944
***** Results of Logistic Regression using PCA with 3 components on wine dataset *****
Training set mean accuracy: 0.7958
Validation set mean accuracy: 0.7778
***** Results of Logistic Regression using PCA with 4 components on wine dataset *****
Training set mean accuracy: 0.9648
Validation set mean accuracy: 0.9167
***** Results of Logistic Regression using PCA with 5 components on wine dataset *****
Training set mean accuracy: 0.9648
Validation set mean accuracy: 0.9167
***** Results of Logistic Regression using PCA with 6 components on wine dataset *****
Training set mean accuracy: 0.9859
Validation set mean accuracy: 0.9722
***** Results of Logistic Regression using PCA with 7 components on wine dataset *****
Training set mean accuracy: 0.9859
Validation set mean accuracy: 0.9722
***** Results of Logistic Regression using PCA with 8 components on wine dataset *****
Training set mean accuracy: 0.9859
Validation set mean accuracy: 0.9444
***** Results of Logistic Regression using PCA with 9 components on wine dataset *****
Training set mean accuracy: 0.9859
Validation set mean accuracy: 0.9444
***** Results of Logistic Regression using PCA with 10 components on wine dataset *****
Training set mean accuracy: 0.9859
Validation set mean accuracy: 0.9444
***** Results of Logistic Regression using PCA with 11 components on wine dataset *****
Training set mean accuracy: 0.9859
Validation set mean accuracy: 0.9444
***** Results of Logistic Regression using PCA with 12 components on wine dataset *****
```

```
Training set mean accuracy: 0.9859
Validation set mean accuracy: 0.9444
***** Results of Logistic Regression using PCA with 13 components on wine dataset *****
Training set mean accuracy: 0.9859
Validation set mean accuracy: 0.9722
```

3. List any collaborators.

```
Collaborators: None.
```

Import packages

[1]:
```
import numpy as np
import sklearn.datasets as skdata
import sklearn.metrics as skmetrics
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt
import warnings

warnings.filterwarnings(action='ignore')
np.random.seed = 1
```

Load datasets

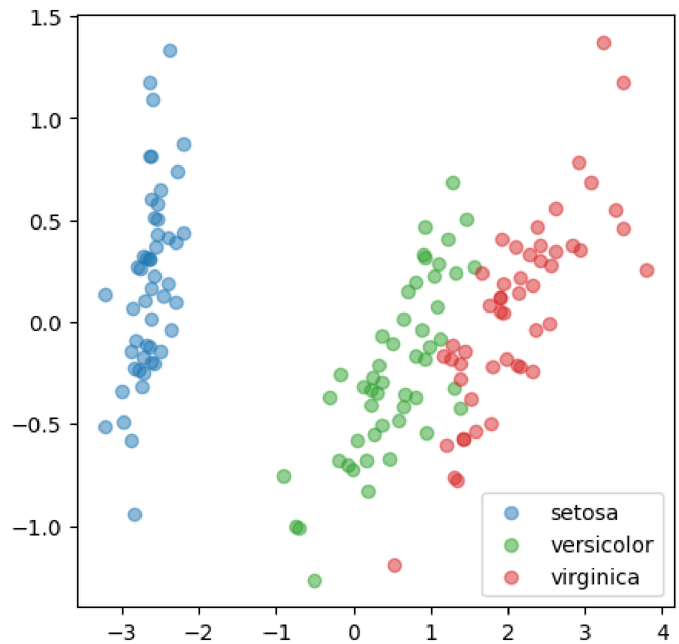
[2]:
```
# Load datasets
datasets = [
    skdata.load_iris(),
    skdata.load_wine()
]

dataset_names = [
    'iris',
    'wine'
]

# Set colors
colors = [
    'tab:blue',
    'tab:green',
    'tab:red'
]
```

Perform PCA on datasets and visualize

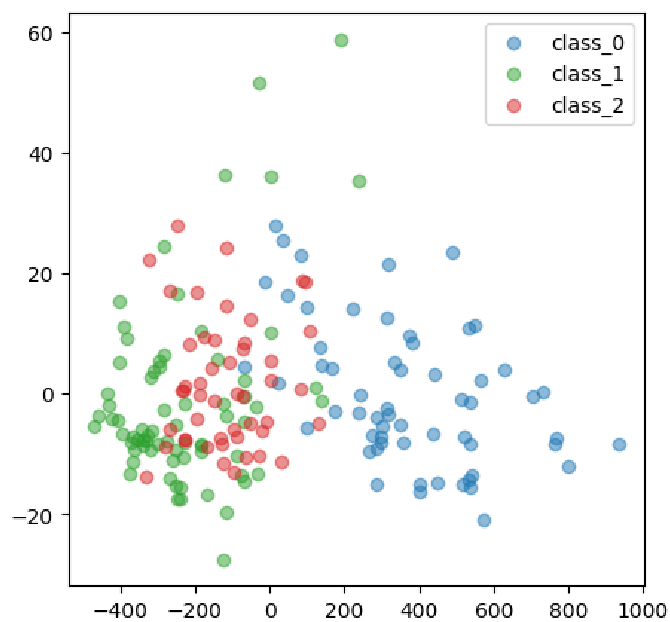
[7]:
```
# Zip up all dataset options
dataset_options = zip(
    datasets,
    dataset_names)
```

```python
for dataset, dataset_name in dataset_options:

    X = dataset.data
    y = dataset.target
    names = dataset.target_names

    n_dim = X.shape[-1]

    # DONE: Instantiate PCA with 2 components (dimensions)
    pca = PCA(n_components=2)

    # DONE: Fit PCA to data
    pca.fit(X)

    # DONE: Use PCA to project (transform) all data points to lower dimensions
    Z = pca.transform(X)

    # DONE: Create figure
    fig = plt.figure(figsize=(5, 5))

    # DONE: Create super title 'Visualization of {} dataset projected from {}␣
↪dimensions to a 2-dimensional subspace'
    fig.suptitle(f'Visualization of {dataset_name} dataset projected from␣
↪{n_dim} dimensions to a 2-dimensional subspace')

    # DONE: Instantiate axis for subplot of a 1 x 1 figure
    ax = fig.add_subplot(1, 1, 1)

    # Iterate through each class and plot them into the figure as scatter plot␣
↪with different colors
    for label, color, name in zip(np.sort(np.unique(y)), colors, names):

        # DONE: Select from projected points the ones belonging to current class
        idx = y == label
        Z_label = Z[idx]

        # DONE: Plot using scatter for selected points with associated color
        # set the points label as name, set alpha to 0.5
        ax.scatter(Z_label[:, 0], Z_label[:, 1], color=color, label=name,␣
↪alpha=0.5)

    # DONE: Turn on legend and set loc to best
    ax.legend(loc='best')
```

Visualization of iris dataset projected from 4 dimensions to a 2-dimensional subspace



Visualization of wine dataset projected from 13 dimensions to a 2-dimensional subspace



Test generalization of the learned subspace through reconstruction loss

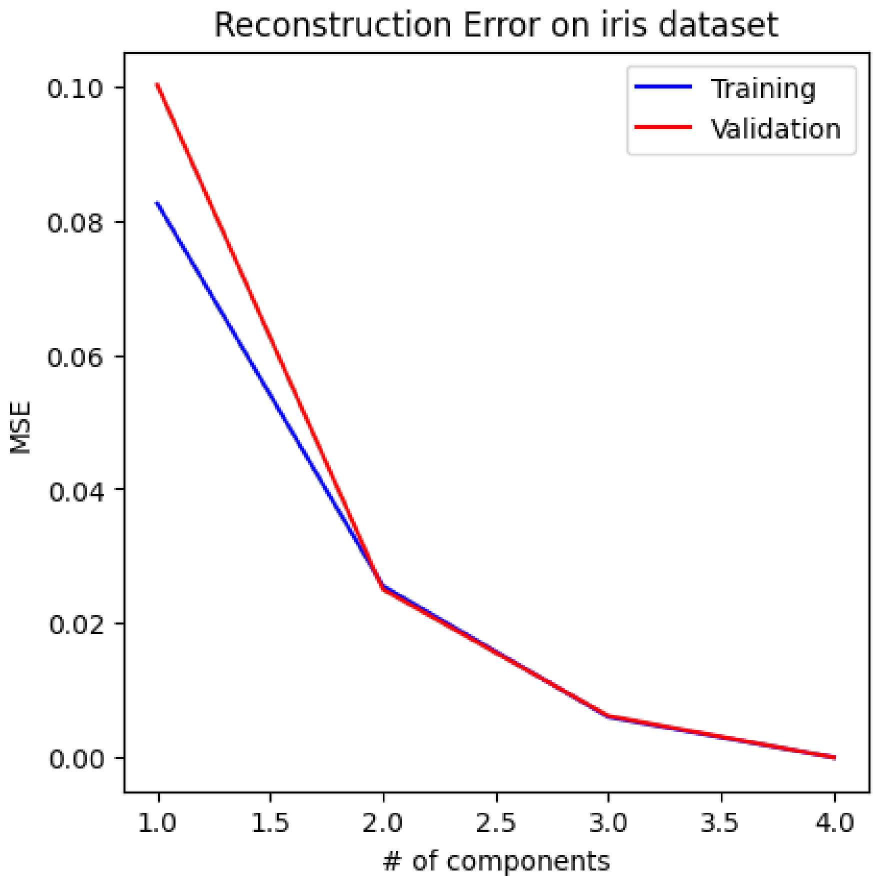
[15]:

```
# Number of dimensions of subspace
dataset_components_list = [
    # For iris dataset
    range(1, 5),
    # For wine dataset
    range(1, 14)
]

# Zip up all dataset options
dataset_options = zip(
    datasets,
    dataset_components_list,
    dataset_names)

for dataset, dataset_components, dataset_name in dataset_options:

    X = dataset.data
    y = dataset.target

    # Shuffle the dataset based on sample indices
    shuffled_indices = np.random.permutation(X.shape[0])

    # Choose the first 80% as training set and the next 20% as validation
    train_split_idx = int(0.80 * X.shape[0])

    train_indices = shuffled_indices[0:train_split_idx]
    val_indices = shuffled_indices[train_split_idx:]

    # Select the examples from X and y to construct our training, validation,␣
↪testing sets
    X_train, y_train = X[train_indices, :], y[train_indices]
    X_val, y_val = X[val_indices, :], y[val_indices]

    # Define empty lists to hold scores for training and validation
    mse_scores_train = []
    mse_scores_val = []

    for components in dataset_components:

        print('***** Fitting PCA with {} components on {} dataset *****'.
↪format(components, dataset_name))

        # DONE: Instantiate PCA with specified components (dimensions)
        pca = PCA(n_components=components)

        # DONE: Fit PCA to training data
        pca.fit(X_train)
```

```
        # DONE?: Project the training data, reconstruct them, and measure loss
        Z_train = pca.transform(X_train)
        X_hat_train = pca.inverse_transform(Z_train)

        mse_score_train = skmetrics.mean_squared_error(X_train, X_hat_train)
        print('Training set mean squared error: {:.4f}'.format(mse_score_train))

        # DONE: Project the validation data, reconstruct them, and measure loss
        Z_val = pca.transform(X_val)
        X_hat_val = pca.inverse_transform(Z_val)

        mse_score_val =  skmetrics.mean_squared_error(X_val, X_hat_val)
        print('Validation set mean squared error: {:.4f}'.format(mse_score_val))

        # DONE: Append training and validation scores to lists of training and␣
↪validation scores
        mse_scores_train.append(mse_score_train)
        mse_scores_val.append(mse_score_val)


    # DONE: Create figure with figsize=(5, 5)
    fig = plt.figure(figsize=(5, 5))

    # DONE: Instantiate axis for subplot of a 1 x 1 figure
    ax = fig.add_subplot(1, 1, 1)

    # DONE: Plot the the number of components on the x-axis and training mse␣
↪scores on the y-axis with color='blue', label='Training'
    ax.plot(dataset_components, mse_scores_train, color='blue',␣
↪label='Training')


    # DONE: Plot the the number of components on the x-axis and validation mse␣
↪scores on the y-axis with color='red', label='Validation'
    ax.plot(dataset_components, mse_scores_val, color='red', label='Validation')


    # DONE: Set title to 'Reconstrction Error on {} dataset'
    ax.set_title(f'Reconstruction Error on {dataset_name} dataset')


    # DONE: Set xlabel to '# of components'
    ax.set_xlabel('# of components')


    # DONE: Set ylabel to 'MSE'
```

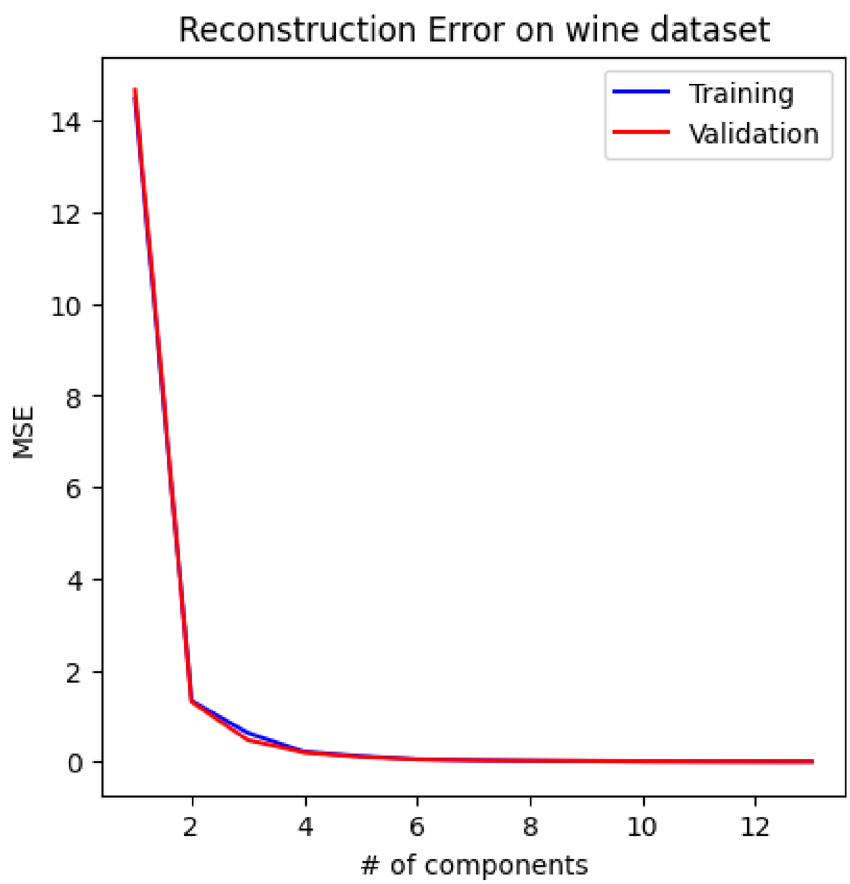
```python
    ax.set_ylabel('MSE')


    # DONE: Set legend with loc='upper right'
    ax.legend(loc='upper right')


    plt.show()
    print('')
    print('')
```

```
***** Fitting PCA with 1 components on iris dataset *****
Training set mean squared error: 0.0826
Validation set mean squared error: 0.1002
***** Fitting PCA with 2 components on iris dataset *****
Training set mean squared error: 0.0256
Validation set mean squared error: 0.0251
***** Fitting PCA with 3 components on iris dataset *****
Training set mean squared error: 0.0060
Validation set mean squared error: 0.0062
***** Fitting PCA with 4 components on iris dataset *****
Training set mean squared error: 0.0000
Validation set mean squared error: 0.0000
```




```
***** Fitting PCA with 1 components on wine dataset *****
Training set mean squared error: 14.4902
Validation set mean squared error: 14.6765
***** Fitting PCA with 2 components on wine dataset *****
Training set mean squared error: 1.3273
Validation set mean squared error: 1.3065
***** Fitting PCA with 3 components on wine dataset *****
Training set mean squared error: 0.6279
Validation set mean squared error: 0.4718
***** Fitting PCA with 4 components on wine dataset *****
Training set mean squared error: 0.2152
Validation set mean squared error: 0.2000
***** Fitting PCA with 5 components on wine dataset *****
Training set mean squared error: 0.1211
Validation set mean squared error: 0.1037
***** Fitting PCA with 6 components on wine dataset *****
Training set mean squared error: 0.0538
```

```
Validation set mean squared error: 0.0486
***** Fitting PCA with 7 components on wine dataset *****
Training set mean squared error: 0.0310
Validation set mean squared error: 0.0312
***** Fitting PCA with 8 components on wine dataset *****
Training set mean squared error: 0.0191
Validation set mean squared error: 0.0216
***** Fitting PCA with 9 components on wine dataset *****
Training set mean squared error: 0.0102
Validation set mean squared error: 0.0131
***** Fitting PCA with 10 components on wine dataset *****
Training set mean squared error: 0.0047
Validation set mean squared error: 0.0073
***** Fitting PCA with 11 components on wine dataset *****
Training set mean squared error: 0.0022
Validation set mean squared error: 0.0025
***** Fitting PCA with 12 components on wine dataset *****
Training set mean squared error: 0.0006
Validation set mean squared error: 0.0006
***** Fitting PCA with 13 components on wine dataset *****
Training set mean squared error: 0.0000
Validation set mean squared error: 0.0000
```



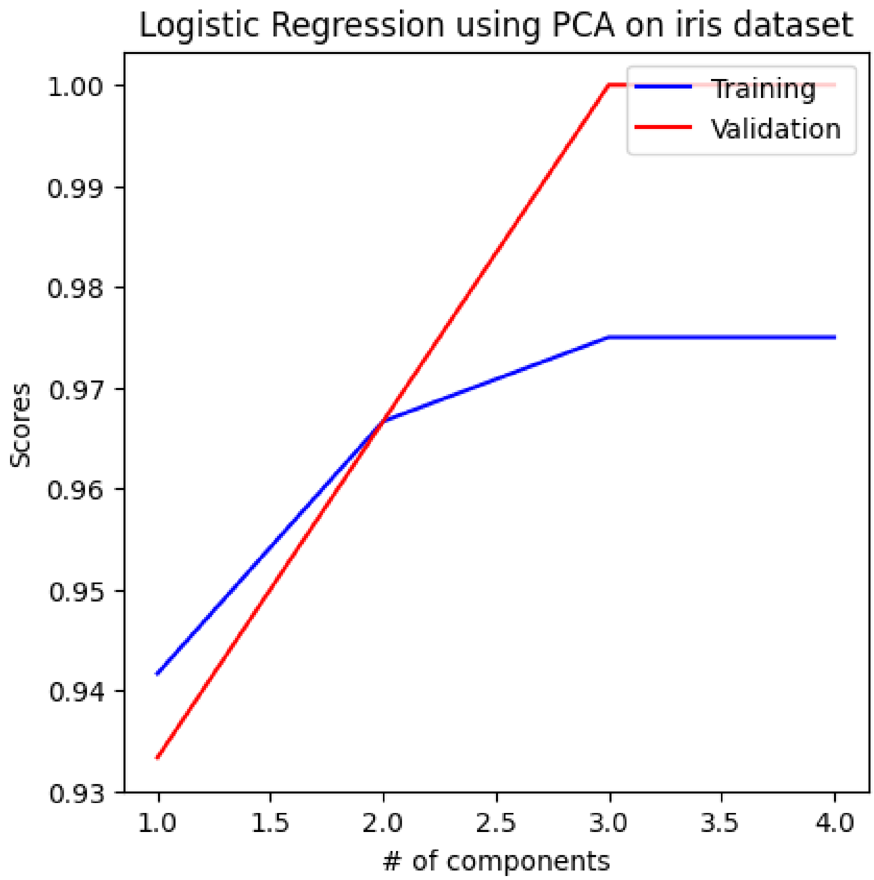
Use PCA as a feature extractor for logistic regression

[36]:
```
# Number of dimensions of subspace
dataset_components_list = [
    # For iris dataset
    range(1, 5),
    # For wine dataset
    range(1, 14)
]

# Zip up all dataset options
dataset_options = zip(
    datasets,
    dataset_components_list,
    dataset_names)

for dataset, dataset_components, dataset_name in dataset_options:

    X = dataset.data
    y = dataset.target

    # Shuffle the dataset based on sample indices
    shuffled_indices = np.random.permutation(X.shape[0])

    # Choose the first 80% as training set and the next 20% as validation
    train_split_idx = int(0.80 * X.shape[0])

    train_indices = shuffled_indices[0:train_split_idx]
    val_indices = shuffled_indices[train_split_idx:]

    # Select the examples from X and y to construct our training, validation,␣
 ↪testing sets
    X_train, y_train = X[train_indices, :], y[train_indices]
    X_val, y_val = X[val_indices, :], y[val_indices]

    # Define empty lists to hold scores for training and validation
    scores_train = []
    scores_val = []

    for components in dataset_components:
```

```
        print('***** Results of Logistic Regression using PCA with {}␣
↪components on {} dataset *****'.format(components, dataset_name))

        # DONE: Instantiate PCA with specified components (dimensions)
        pca = PCA(n_components=components)

        # DONE: Fit PCA to training data
        pca.fit(X_train)


        # DONE: Project the training data
        Z_train = pca.transform(X_train)

        # DONE: Instantiate LogisticRegression with tol=1e-4
        logistic = LogisticRegression(tol=1e-4)

        # DONE: Train model using projected training data
        logistic.fit(Z_train, y_train)


        # DONE: Score model using mean accuracy on training set
        predictions_train = logistic.predict(Z_train)
        score_train = skmetrics.accuracy_score(y_train, predictions_train)
        print('Training set mean accuracy: {:.4f}'.format(score_train))

        # DONE: Project the validation data and test model on it
        Z_val = pca.transform(X_val)

        # DONE: Score model using mean accuracy validation set
        predictions_val = logistic.predict(Z_val)
        score_val = skmetrics.accuracy_score(y_val, predictions_val)
        print('Validation set mean accuracy: {:.4f}'.format(score_val))

        # DONE: Append training and validation scores to lists of training and␣
↪validation scores
        scores_train.append(score_train)
        scores_val.append(score_val)


    # DONE: Create figure with figsize=(5, 5)
    fig = plt.figure(figsize=(5, 5))

    # DONE: Instantiate axis for subplot of a 1 x 1 figure
    ax = fig.add_subplot(1, 1, 1)

    # DONE: Plot the the number of components on the x-axis and training scores␣
↪on the y-axis with color='blue', label='Training'
```

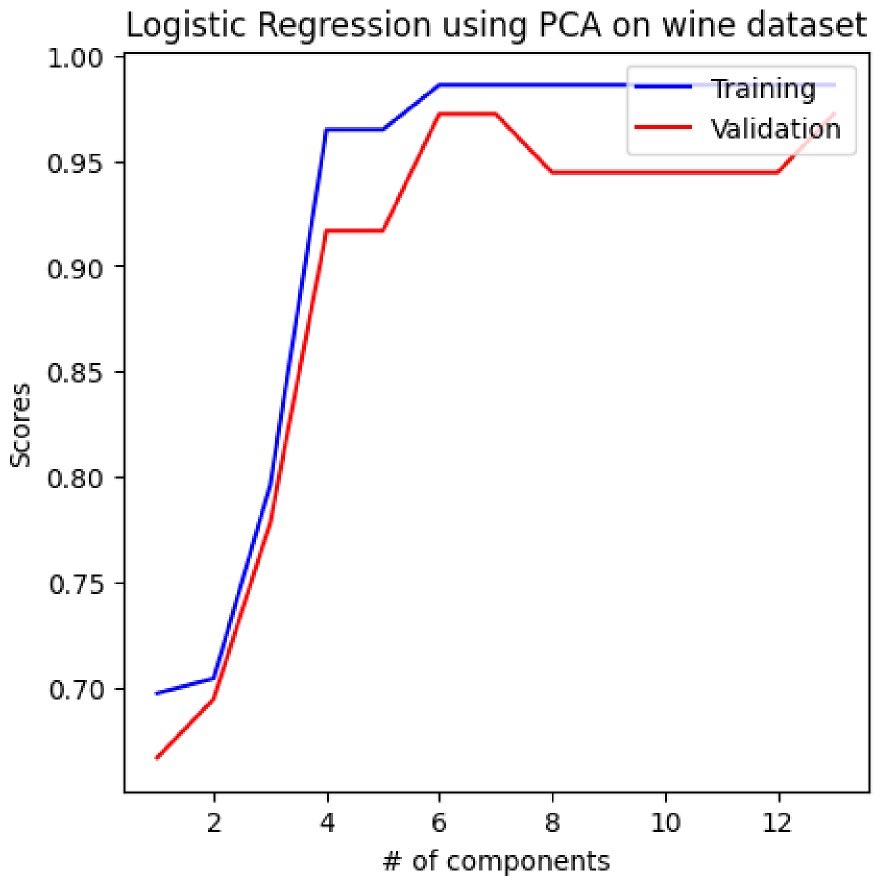
```
    ax.plot(dataset_components, scores_train, color='blue', label='Training')

    # DONE: Plot the the number of components on the x-axis and validation␣
↪scores on the y-axis with color='red', label='Validation'
    ax.plot(dataset_components, scores_val, color='red', label='Validation')

    # DONE: Set title to 'Logistic Regression using PCA on {} dataset'
    ax.set_title(f'Logistic Regression using PCA on {dataset_name} dataset')

    # DONE: Set xlabel to '# of components'
    ax.set_xlabel('# of components')

    # DONE: Set ylabel to 'Scores'
    ax.set_ylabel('Scores')

    # DONE: Set legend with loc='upper right'
    ax.legend(loc='upper right')

    plt.show()
    print('')
```

```
***** Results of Logistic Regression using PCA with 1 components on iris dataset
*****
Training set mean accuracy: 0.9417
Validation set mean accuracy: 0.9333
***** Results of Logistic Regression using PCA with 2 components on iris dataset
*****
Training set mean accuracy: 0.9667
Validation set mean accuracy: 0.9667
***** Results of Logistic Regression using PCA with 3 components on iris dataset
*****
Training set mean accuracy: 0.9750
Validation set mean accuracy: 1.0000
***** Results of Logistic Regression using PCA with 4 components on iris dataset
*****
Training set mean accuracy: 0.9750
Validation set mean accuracy: 1.0000
```




```
***** Results of Logistic Regression using PCA with 1 components on wine dataset
*****
Training set mean accuracy: 0.6972
Validation set mean accuracy: 0.6667
***** Results of Logistic Regression using PCA with 2 components on wine dataset
*****
Training set mean accuracy: 0.7042
Validation set mean accuracy: 0.6944
***** Results of Logistic Regression using PCA with 3 components on wine dataset
*****
Training set mean accuracy: 0.7958
Validation set mean accuracy: 0.7778
***** Results of Logistic Regression using PCA with 4 components on wine dataset
*****
Training set mean accuracy: 0.9648
Validation set mean accuracy: 0.9167
***** Results of Logistic Regression using PCA with 5 components on wine dataset
*****
```

```
Training set mean accuracy: 0.9648
Validation set mean accuracy: 0.9167
***** Results of Logistic Regression using PCA with 6 components on wine dataset
*****
Training set mean accuracy: 0.9859
Validation set mean accuracy: 0.9722
***** Results of Logistic Regression using PCA with 7 components on wine dataset
*****
Training set mean accuracy: 0.9859
Validation set mean accuracy: 0.9722
***** Results of Logistic Regression using PCA with 8 components on wine dataset
*****
Training set mean accuracy: 0.9859
Validation set mean accuracy: 0.9444
***** Results of Logistic Regression using PCA with 9 components on wine dataset
*****
Training set mean accuracy: 0.9859
Validation set mean accuracy: 0.9444
***** Results of Logistic Regression using PCA with 10 components on wine
dataset *****
Training set mean accuracy: 0.9859
Validation set mean accuracy: 0.9444
***** Results of Logistic Regression using PCA with 11 components on wine
dataset *****
Training set mean accuracy: 0.9859
Validation set mean accuracy: 0.9444
***** Results of Logistic Regression using PCA with 12 components on wine
dataset *****
Training set mean accuracy: 0.9859
Validation set mean accuracy: 0.9444
***** Results of Logistic Regression using PCA with 13 components on wine
dataset *****
Training set mean accuracy: 0.9859
Validation set mean accuracy: 0.9722
```


Logistic Regression using PCA on wine dataset
Training
Validation
Scores
# of components
1.00
0.95
0.90
0.85
0.80
0.75
0.70
2
4
6
8
10
12

```
[ ]:
```