EXPERIMENT 4 – DATA REDUCTION

AIM: Perform dimensionality reduction to obtain a reduced or "compressed" representation of the original data using principal components analysis.

SOFTWARE REQUIRED:

Spyder IDE 5.1.5

Anaconda3 2021.11 (Python 3.9.7 64-bit)

Anaconda Inc., 2021.11

DATA SET: Real Estate Data Set

PYTHON CODE:

```
import matplotlib.pyplot as plt # Provides an implicit way of plotting
import numpy as np # Support for large, multi-dimensional arrays and matrices
from numpy.linalg import eig # Compute the eigenvalues and right eigenvectors
of a square array
import pandas as pd  # Library for working with data sets
import seaborn as sns # Provides high-level API to visualize data

"""
There is a Unicode character '\u0332', COMBINING LOW LINE*, which acts as an
underline on the character that precedes it in a string. The center() method
will center align the string, using a specified character (space is the
default) as the fill character.
"""

def print_csv_file(df, heading):
    print("\n")
    print('{:s}'.format('\u0332'.join(heading.center(100))))
    print(df)

def standardize_the_data_set():

    print("\nMean:")

    mean_X2 = df['X2'].mean()
    mean_X3 = df['X3'].mean()
    mean_X4 = df['X4'].mean()
    mean_Y = df['Y'].mean()
```

```
    print("X2. The age of house in years: " + str(mean_X2))
    print("X3. The distance to nearest MRT station in meters: " +
str(mean_X3))
    print("X4. The number of convenience stores within walking distance: " +
str(mean_X4))
    print("Y. House price per local unit area: " + str(mean_Y))

    print("\nStandard Deviation:")

    std_X2 = df['X2'].std()
    std_X3 = df['X3'].std()
    std_X4 = df['X4'].std()
    std_Y = df['Y'].std()

    print("X2. The age of house in years: " + str(std_X2))
    print("X3. The distance to nearest MRT station in meters: " + str(std_X3))
    print("X4. The number of convenience stores within walking distance: " +
str(std_X4))
    print("Y. House price per local unit area: " + str(std_Y))

    heading = "Step 1 - Standardize the dataset (Z-Score Normalization)"
    for i in df.index:
        df.loc[i, 'X2'] = ((df.loc[i, 'X2']) - mean_X2)/std_X2
        df.loc[i, 'X3'] = ((df.loc[i, 'X3']) - mean_X3)/std_X3
        df.loc[i, 'X4'] = ((df.loc[i, 'X4']) - mean_X4)/std_X4
        df.loc[i, 'Y'] = ((df.loc[i, 'Y']) - mean_Y)/std_Y
    print_csv_file(df, heading)

    standardized_dataset = df
    return standardized_dataset

# Calculate the covariance matrix for the whole dataset:-
def covariance_matrix():

    size = df.shape[1] # Get the shape of the DataFrame, which is a tuple
where the first element is the number of rows and the second is the number of
columns.
    covariance_matrix_1D = [0] * (size * size)

    sum = 0
    for i in df.index:
        sum = sum + df.loc[i, 'X2'] * df.loc[i, 'X2']
    covariance_matrix_1D[0] = sum / len(df)  # var(f1)
    sum = 0
    for i in df.index:
        sum = sum + df.loc[i, 'X2'] * df.loc[i, 'X3']
    covariance_matrix_1D[1] = sum / len(df)  # cov(f1,f2)
    sum = 0
```

```
for i in df.index:
    sum = sum + df.loc[i, 'X2'] * df.loc[i, 'X4']
covariance_matrix_1D[2] = sum / len(df)  # cov(f1,f3)
sum = 0
for i in df.index:
    sum = sum + df.loc[i, 'X2'] * df.loc[i, 'Y']
covariance_matrix_1D[3] = sum / len(df)  # cov(f1,f4)

sum = 0
for i in df.index:
    sum = sum + df.loc[i, 'X3'] * df.loc[i, 'X2']
covariance_matrix_1D[4] = sum / len(df)  # cov(f2,f1)
sum = 0
for i in df.index:
    sum = sum + df.loc[i, 'X3'] * df.loc[i, 'X3']
covariance_matrix_1D[5] = sum / len(df)  # var(f2)
sum = 0
for i in df.index:
    sum = sum + df.loc[i, 'X3'] * df.loc[i, 'X4']
covariance_matrix_1D[6] = sum / len(df)  # cov(f2,f3)
sum = 0
for i in df.index:
    sum = sum + df.loc[i, 'X3'] * df.loc[i, 'Y']
covariance_matrix_1D[7] = sum / len(df)  # cov(f2,f4)

sum = 0
for i in df.index:
    sum = sum + df.loc[i, 'X4'] * df.loc[i, 'X2']
covariance_matrix_1D[8] = sum / len(df)  # cov(f3,f1)
sum = 0
for i in df.index:
    sum = sum + df.loc[i, 'X4'] * df.loc[i, 'X3']
covariance_matrix_1D[9] = sum / len(df)  # cov(f3,f2)
sum = 0
for i in df.index:
    sum = sum + df.loc[i, 'X4'] * df.loc[i, 'X4']
covariance_matrix_1D[10] = sum / len(df)  # var(f3)
sum = 0
for i in df.index:
    sum = sum + df.loc[i, 'X4'] * df.loc[i, 'Y']
covariance_matrix_1D[11] = sum / len(df)  # cov(f3,f4)

sum = 0
for i in df.index:
    sum = sum + df.loc[i, 'Y'] * df.loc[i, 'X2']
covariance_matrix_1D[12] = sum / len(df)  # cov(f4,f1)
sum = 0
for i in df.index:
```

```
        sum = sum + df.loc[i, 'Y'] * df.loc[i, 'X3']
    covariance_matrix_1D[13] = sum / len(df)  # cov(f4,f2)
    sum = 0
    for i in df.index:
        sum = sum + df.loc[i, 'Y'] * df.loc[i, 'X4']
    covariance_matrix_1D[14] = sum / len(df)  # cov(f4,f3)
    sum = 0
    for i in df.index:
        sum = sum + df.loc[i, 'Y'] * df.loc[i, 'Y']
    covariance_matrix_1D[15] = sum / len(df)  # var(f4)

    covariance_matrix_multidim = np.reshape(covariance_matrix_1D, (size,
size))
    print("\nThe covariance matrix is as follows:\n{}".format(
        covariance_matrix_multidim))
    return covariance_matrix_multidim

# Determine explained variance:-
def determine_explained_variance(eigenvalues):

    total_eigenvalues = sum(eigenvalues)
    explained_variance = [(i/total_eigenvalues) for i in sorted(eigenvalues,
reverse=True)]
    cumulative_explained_variance = np.cumsum(explained_variance) # Return the
cumulative sum of the elements along a given axis

    plt.bar(range(0,len(explained_variance)), explained_variance, alpha=0.5,
align='center', label='Individual Explained Variance')
    plt.step(range(0,len(cumulative_explained_variance)),
cumulative_explained_variance, where='mid',label='Cumulative Explained
Variance')

    plt.xlabel('Principal Component Index')
    plt.ylabel('Explained Variance Ratio')
    plt.legend(loc='best')
    plt.grid(True)
    plt.tight_layout()
    plt.show()

# Convert CSV data into multidimensional array:-
def convert_csv_array():
    size = df.shape[1]
    single_dimensional_array = []
    for i in df.index:
        single_dimensional_array.append(df.loc[i, 'X2'])
        single_dimensional_array.append(df.loc[i, 'X3'])
        single_dimensional_array.append(df.loc[i, 'X4'])
        single_dimensional_array.append(df.loc[i, 'Y'])
```

```
    multi_dimensional_array = np.reshape(single_dimensional_array, (len(df),
size))
    return multi_dimensional_array

# Driver Code: main() ; Execution starts here.

print("\nThere are three regressor variables and one response variable
(namely, y):")
print("X2 - Age of House in year(s)")
print("X3 - Distance to Nearest MRT station in meter(s)")
print("X4 - Number of Convenience Stores within Walking Distance")
print("Y - House Price per Local Unit Area")

heading = "Original Data Set"
df = pd.read_csv("Real Estate Data Set.csv")
print_csv_file(df, heading)

# Step 1 - Standardize the dataset:-
standardized_dataset = standardize_the_data_set()
print("\nSince we have standardized the dataset, so the mean for each feature
is 0 and the standard deviation is 1.\n")

heading = "Step 2 - Calculate the covariance matrix for the features in the
dataset"
print('{:s}'.format('\u0332'.join(heading.center(100))))
covariance_matrix_multidim = covariance_matrix()

print("\nNow, we will check the co-relation between our scaled dataset using a
heat map.  The correlation between various features is given by the corr()
function and then the heat map is plotted by the heatmap() function. The
colour scale on the side of the heatmap helps determine the magnitude of the
co-relation.")

print("\nIn our example, we can see that a darker shade represents less co-
relation while a lighter shade represents more co-relation. The diagonal of
the heatmap represents the co-relation of a feature with itself – which is
always 1.0, thus, the diagonal of the heatmap is of the highest shade.")

sns.heatmap(standardized_dataset.corr())
plt.title("Co-relation between features before Principal Component Analysis
(PCA)")
plt.grid(True)
plt.tight_layout()
plt.show()

heading = "Step 3 - Calculate and sort the eigenvalues and eigenvectors for
the covariance matrix"
print("\n")
```

```
print('{:s}'.format('\u0332'.join(heading.center(100))))
eigenvalues, eigen_vectors = eig(covariance_matrix_multidim)

index = eigenvalues.argsort()[::-1] # Returns the indices that would sort an
array
eigenvalues = eigenvalues[index]
eigen_vectors = eigen_vectors[:, index]

print("\nSorted Eigenvalues: ", eigenvalues)
print("\nCorresponding Eigen Vectors:\n{}".format(eigen_vectors))
determine_explained_variance(eigenvalues)

heading = "Step 4 - Pick k eigenvalues and form a matrix of eigenvectors"
print("\n")
print('{:s}'.format('\u0332'.join(heading.center(100))))

eigenvectors_1D = eigen_vectors.flatten() # Return a copy of the array
(matrix) collapsed into one dimension.

while True:
    top = int(input("Enter the number of top eigen vectors to transform the
original matrix: "))
    if top > df.shape[1]:
        print("Invalid Input! Please try again.")
    else:
        break

top_eigenvectors_1D = []  # NULL Single Dimensional Array Declaration
for i in range(int(len(eigenvectors_1D)/len(eigen_vectors)) * top):
    top_eigenvectors_1D.append(eigenvectors_1D[i])
top_eigenvectors_multidim = np.reshape(top_eigenvectors_1D,
(len(eigen_vectors), top))
print("\nIf we choose the top " + str(top) + " eigenvectors, the matrix will
look like this:\n{}".format(top_eigenvectors_multidim))

heading = "Step 5 - Tranform the Original Matrix"
print("\n")
print('{:s}'.format('\u0332'.join(heading.center(100))))

feature_matrix = convert_csv_array()
transformed_data = np.dot(feature_matrix, top_eigenvectors_multidim) # Matrix
Multiplication

print("\nThe transformed data is as follows:\n")
for i in transformed_data:
    print(i)
```

```
print("\nNow that we have applied PCA and obtained the reduced feature set, we
will check the co-relation between various Principal Components, again by
using a heatmap.")

new_dataframe = pd.DataFrame(transformed_data)
sns.heatmap(new_dataframe.corr())
plt.title("Co-relation between features after Principal Component Analysis
(PCA)")
plt.grid(True)
plt.tight_layout()
plt.show()
```

PLOTS:

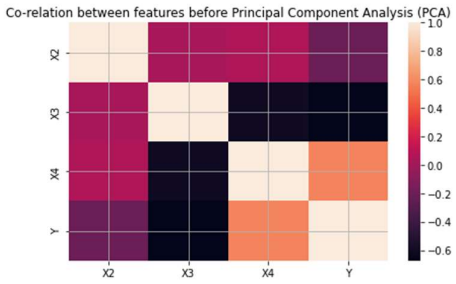

Figure 1. Co-relation between features before Principal Component Analysis (PCA)

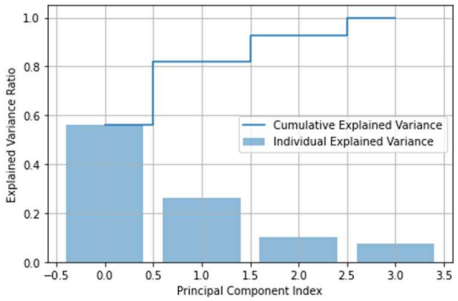

Figure 2. Determine explained variance

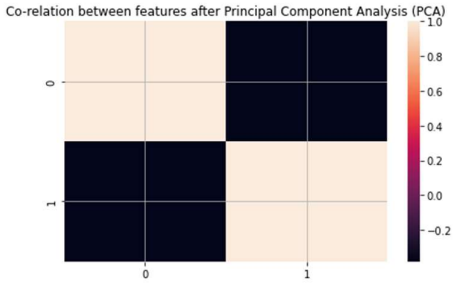

Figure 3. Co-relation between features after Principal Component Analysis (PCA)

RESULT:

Thus, demonstrated dimensionality reduction by reducing the number of random variables or attributes under consideration using principal component analysis, which transforms or projects the original data onto a smaller space. All simulation results were verified successfully.

```
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.29.0 -- An enhanced Interactive Python.

Restarting kernel...
```

```
In [1]:         'E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester IV/
19CCE213 - Machine Learning and Artificial Intelligence/Lab/Experiment 4 - Principal
Component Analysis/Expt_4_Code.py'      ='E:/Plan B/Amrita Vishwa Vidyapeetham/Subject
Materials/Semester IV/19CCE213 - Machine Learning and Artificial Intelligence/Lab/
Experiment 4 - Principal Component Analysis'

There are three regressor variables and one response variable (namely, y):
X2 - Age of House in year(s)
X3 - Distance to Nearest MRT station in meter(s)
X4 - Number of Convenience Stores within Walking Distance
Y - House Price per Local Unit Area


                                        Original Data Set
       X2          X3  X4     Y
0    32.0    84.87882  10  37.9
1    19.5   306.59470   9  42.2
2    13.3   561.98450   5  47.3
3    13.3   561.98450   5  54.8
4     5.0   390.56840   5  43.1
..    ...         ...  ..   ...
409  13.7  4082.01500   0  15.4
410   5.6    90.45606   9  50.0
411  18.8   390.96960   7  40.6
412   8.1   104.81010   5  52.5
413   6.5    90.45606   9  63.9

[414 rows x 4 columns]

Mean:
X2. The age of house in years: 17.71256038647343
X3. The distance to nearest MRT station in meters: 1083.8856889130436
X4. The number of convenience stores within walking distance: 4.094202898550725
Y. House price per local unit area: 37.98019323671498

Standard Deviation:
X2. The age of house in years: 11.392484533242536
X3. The distance to nearest MRT station in meters: 1262.1095954078514
X4. The number of convenience stores within walking distance: 2.945561805663617
Y. House price per local unit area: 13.606487697735314


                    Step 1 - Standardize the dataset (Z-Score Normalization)
          X2        X3        X4         Y
0    1.254111 -0.791537  2.004982 -0.005894
1    0.156896 -0.615866  1.665488  0.310132
```

```
2   -0.387322 -0.413515  0.307513  0.684953
3   -0.387322 -0.413515  0.307513  1.236161
4   -1.115873 -0.549332  0.307513  0.376277
..        ...       ...       ...       ...
409 -0.352211  2.375490 -1.389957 -1.659517
410 -1.063206 -0.787118  1.665488  0.883388
411  0.095452 -0.549014  0.986500  0.192541
412 -0.843763 -0.775745  0.307513  1.067124
413 -0.984207 -0.787118  1.665488  1.904959

[414 rows x 4 columns]
```

Since we have standardized the dataset, so the mean for each feature is 0 and the standard deviation is 1.

Step 2 - Calculate the covariance matrix for the features in the dataset

```
The covariance matrix is as follows:
[[ 0.99758454  0.02556016  0.04947272 -0.21005843]
 [ 0.02556016  0.99758454 -0.60106378 -0.67198577]
 [ 0.04947272 -0.60106378  0.99758454  0.56962567]
 [-0.21005843 -0.67198577  0.56962567  0.99758454]]
```

Now, we will check the co-relation between our scaled dataset using a heat map. The correlation between various features is given by the corr() function and then the heat map is plotted by the heatmap() function. The colour scale on the side of the heatmap helps determine the magnitude of the co-relation.

In our example, we can clearly see that a darker shade represents less co-relation while a lighter shade represents more co-relation. The diagonal of the heatmap represents the co-relation of a feature with itself – which is always 1.0, thus, the diagonal of the heatmap is of the highest shade.

Step 3 - Calculate and sort the eigenvalues and eigenvectors for the covariance matrix

```
Sorted Eigenvalues:  [2.23691337 1.04279849 0.41043771 0.3001886 ]

Corresponding Eigen Vectors:
[[ 0.08954605  0.955056   -0.20221334 -0.19738113]
 [ 0.58700983 -0.09294975  0.46288628 -0.65765956]
 [-0.55073343  0.22961701  0.80147479  0.04008594]
 [-0.58659497 -0.16280176 -0.32013157 -0.72589098]]
```

Step 4 - Pick k eigenvalues and form a matrix of eigenvectors

```
Enter the number of top eigen vectors to transform the original matrix:
```

lighter shade represents more co-relation. The diagonal of the heatmap represents the co-relation of a feature with itself – which is always 1.0, thus, the diagonal of the heatmap is of the highest shade.

Step 3 - Calculate and sort the eigenvalues and eigenvectors for the covariance matrix

Sorted Eigenvalues: [2.23691337 1.04279849 0.41043771 0.3001886]

Corresponding Eigen Vectors:
```
[[ 0.08954605  0.955056   -0.20221334 -0.19738113]
 [ 0.58700983 -0.09294975  0.46288628 -0.65765956]
 [-0.55073343  0.22961701  0.80147479  0.04008594]
 [-0.58659497 -0.16280176 -0.32013157 -0.72589098]]
```

Step 4 - Pick k eigenvalues and form a matrix of eigenvectors

Enter the number of top eigen vectors to transform the original matrix: 5
Invalid Input! Please try again.

Enter the number of top eigen vectors to transform the original matrix: 2

If we choose the top 2 eigenvectors, the matrix will look like this:
```
[[ 0.08954605  0.955056  ]
 [-0.20221334 -0.19738113]
 [ 0.58700983 -0.09294975]
 [ 0.46288628 -0.65765956]]
```

Step 5 - Tranform the Original Matrix

The transformed data is as follows:

```
[1.44657583 1.17149427]
[ 1.25979938 -0.08736267]
[ 0.54650337 -0.7673434 ]
[ 0.80164982 -1.12985038]
[ 0.36584657 -1.23333825]
[-0.67633824 -0.74157388]
[0.86372155 1.27550922]
[ 0.82435962 -0.14016339]
[-1.86866245  1.50477925]
[-0.86886282  0.70843689]
[-0.25724691  1.47095995]
[ 1.73158598 -1.9286594 ]
[ 0.28316487 -0.39491023]
[-0.70207758  0.69693689]
[-0.192411   -0.21010549]
[0.23081387 1.04780015]
[ 1.45999214 -2.97382072]
[-0.51902291  0.2392696 ]
[ 1.03361877 -0.28822819]
[ 0.95222839 -1.75477588]
```

```
[-0.80818673 -0.83997287]
[ 1.11466317 -1.22879345]
[-1.13975971  0.44860953]
[ 0.98564688 -1.08348952]
[0.27779542 1.89255059]
[-0.76453376  1.5050247 ]
[ 0.79764262 -2.02474936]
[ 0.10338964 -0.30362225]
[ 0.38410791 -0.22597305]
[ 0.84890484 -1.74345953]
[-1.84228024  1.04579897]
[0.28132757 1.5814241 ]
[-0.4911769   2.06580964]
[ 0.87716574 -0.59003273]
[ 1.28407102 -0.9756436 ]
[-1.68916015 -0.14267043]
[-1.09041153  0.40931808]
[-1.13716831  0.18842885]
[ 0.67666457 -1.67581912]
[ 0.57556131 -0.42842139]
[-2.07975688  0.38278439]
[-1.9738878   0.54229112]
[0.30388051 1.75969053]
[0.47046222 1.61566448]
[ 0.49299656 -1.93895561]
[1.03304894 1.5377278 ]
[1.24507448 0.08212413]
[0.79602852 0.49171102]
[-2.16540054  1.31033372]
[-1.91676249  1.7392897 ]
[0.31570199 0.12596965]
[-1.20576799  1.96644124]
[-0.53534227  1.71590704]
[ 0.27191509 -0.35042689]
[ 0.7618827  -0.70264305]
[-1.54427238  2.46335558]
[1.15077581 1.13061485]
[ 1.15995999 -1.87262555]
[-1.58990485  1.36039652]
[ 0.41600606 -0.49517215]
[-1.17331599  0.1770687 ]
[ 1.27225714 -2.19079055]
[-0.74677287  0.31766871]
[ 0.52963203 -2.00050651]
[-1.23484297  0.73804721]
[1.32316845 1.59822884]
[ 0.82380126 -1.93675667]
[ 0.90520821 -1.55741559]
[0.5182437  1.18642902]
[ 0.35991179 -0.57823749]
[ 1.76456158 -1.94701061]
[0.08868027 1.45869743]
[0.94308155 1.30073665]
[-1.9387048   0.20138033]
[ 1.76261057 -1.78467497]
```

```
[-0.99192827  0.01057458]
[-0.04035429  1.68937365]
[-0.79375773  0.69436986]
[-0.44945085  2.26208642]
[-1.05794792  0.62486618]
[ 0.10185788 -0.51873744]
[0.55564673 1.20447767]
[ 1.03736318 -0.81322173]
[-1.1084599   1.39692788]
[ 0.86538534 -0.47760649]
[ 1.39390861 -2.14282629]
[-1.17483446 -0.763787  ]
[-1.96969984  0.5458409 ]
[-0.59599423 -1.14432204]
[-1.6646079   0.73743132]
[-0.37367971 -1.61921628]
[-0.75700873 -0.89486085]
[-0.76854462  0.81037184]
[-1.45876189  2.36635661]
[0.61230694 1.91260656]
[ 0.73118053 -1.35766032]
[ 1.77999933 -1.98794416]
[-0.17431493  1.16680526]
[ 0.74042706 -0.64365943]
[ 1.87185205 -2.11844667]
[ 0.00612145 -0.00683925]
[-0.68245617  0.06587643]
[ 0.95045952 -2.10721024]
[ 0.64347133 -1.78123297]
[0.35391064 1.66597338]
[ 1.49060971 -3.01732156]
[ 1.22788847 -0.46714895]
[-1.09137091  0.14236073]
[-0.36303447  1.47792523]
[-0.82214611 -0.81829183]
[ 0.72516263 -1.33961026]
[0.89206208 1.28516746]
[-0.94887336 -0.06048123]
[-0.56596282  1.27210586]
[1.55145361 0.36367272]
[-0.0527816  -0.04471449]
[-2.24115184  1.61843276]
[-2.19689215  0.50491669]
[-0.48957524  0.94914646]
[ 1.03456701 -1.04256767]
[0.01336669 0.01691351]
[ 0.58385082 -0.76511906]
[0.01033999 1.52829902]
[-0.55537103 -1.57864008]
[ 1.30726037 -1.55943145]
[ 0.75314626 -1.82687151]
[1.03789641 0.59319367]
[ 0.76232125 -1.90806577]
[1.49174841 0.96906196]
[0.98412691 1.64059807]
```

```
[0.94332919 0.98226584]
[-0.7439012  -0.92946064]
[0.33034484 0.83368963]
[ 0.94616787 -0.06149914]
[0.79036245 1.19127954]
[-1.37908639  1.07080828]
[ 0.54202345 -0.87564954]
[ 0.79046883 -0.74059382]
[-0.66610043 -0.76204334]
[ 0.39124135 -0.55796309]
[ 0.75246285 -0.67975956]
[-0.702354   -0.65835695]
[0.25160584 0.23894531]
[ 0.31509656 -0.38327838]
[-1.03249379  0.03020943]
[ 0.41497783 -1.60194262]
[-0.3274402  -1.90247965]
[ 0.93704706 -1.49926872]
[-1.01608454 -0.74670639]
[1.1168802  1.37592687]
[1.22543746 1.05895425]
[ 1.21708464 -1.44484243]
[-1.01469801  0.0144255 ]
[ 0.50438671 -1.03055895]
[-1.88805297  0.42983867]
[-2.08839072  0.41405109]
[-1.41232108  2.14090599]
[-0.06949003  0.02473853]
[ 0.29185097 -0.50120956]
[-0.01297094 -0.1125624 ]
[ 1.50662176 -2.11095324]
[-0.07636325  0.1540415 ]
[-2.20470476  0.79423157]
[ 0.86098283 -1.49458104]
[-0.22538162 -2.04748244]
[-0.92370365  0.09409772]
[ 1.57906048 -3.14299065]
[1.16595582 0.61185771]
[0.32961926 0.91729531]
[-1.32322971 -0.12058823]
[-2.12044606  1.25744114]
[ 0.89002344 -2.10850239]
[ 1.73394401 -1.90350977]
[0.17791328 2.22622812]
[ 0.82371364 -1.46311992]
[-0.07225505  1.24858638]
[-2.04392411  0.17152406]
[1.37921712 1.0736674 ]
[-0.10828137 -0.33130027]
[-0.38530294 -0.33600964]
[-2.04766644  1.45965019]
[ 1.48126093 -1.36388573]
[-0.91076891  0.21011766]
[-1.95437591  0.5276442 ]
[-1.64014314  0.36888183]
```

```
[-0.72934677  1.92170132]
[-0.78721174  0.72306928]
[-1.74834662 -0.14907529]
[1.27088804 1.1435506 ]
[-1.89957294  0.38920626]
[1.13844033 1.21542954]
[-0.40546161 -0.25129824]
[1.10913238 2.02764406]
[ 0.60875765 -1.14383388]
[-1.56163209 -0.08226374]
[0.14555287 0.02155981]
[-0.36407269  0.61808509]
[1.41056368 0.93141526]
[0.89360874 1.36445129]
[0.66644881 0.32487074]
[-1.22882507  0.71999715]
[ 0.49730401 -0.71644185]
[-0.20600699  2.14262955]
[ 0.5674433  -0.47388722]
[-1.05454597  0.62003275]
[-0.87933684  0.36687657]
[1.529853   0.00901584]
[-0.11616555  2.02539579]
[-1.11048079  0.10301196]
[1.15753036 1.31014064]
[ 0.67699629 -1.65641363]
[-0.43831681 -1.52738118]
[-0.76793382  0.27631832]
[ 1.72739801 -1.93220919]
[-0.90811423  0.78320443]
[ 1.24651459 -0.3781891 ]
[-0.32125492  1.77558046]
[0.67054512 0.77721947]
[ 0.44096881 -0.56211515]
[1.03374513 0.82751292]
[ 2.64626783 -0.32962125]
[-0.91768652 -0.67607302]
[1.74070581 0.48763886]
[-0.78762631 -0.85136001]
[0.90617688 1.07824703]
[ 0.76675407 -1.84620521]
[-2.16242313  0.77573372]
[1.10052387 1.12879139]
[-0.90281079 -1.10117003]
[-1.36879483  2.14916147]
[-0.65204847 -1.05996315]
[-2.09897782  0.65988735]
[-1.77632233  1.35873465]
[1.08089243 2.07553465]
[-0.75560028 -0.30784289]
[ 0.71966011 -0.42862134]
[ 1.99352094 -2.41455036]
[-0.76202402 -0.26026139]
[-0.7091648  -0.35436263]
[-1.05512132  0.60039033]
```

```
[-0.88314977 -0.15125537]
[ 0.7973892 -0.4630576]
[-0.65655562 -1.21668227]
[1.3855799  0.78517276]
[-0.94965263 -0.33549988]
[ 0.26734926 -0.95153829]
[ 0.56980868 -0.18825568]
[-1.28617633  1.18720171]
[-1.32727109  1.00669009]
[-2.23285193  0.41575804]
[0.5821079  2.19844708]
[-1.54068169  2.45870638]
[ 1.54473649 -1.70118722]
[-0.83299959  1.61166572]
[ 0.64087031 -1.97131533]
[-1.9178618   1.55568079]
[-0.91233991  0.50805119]
[-0.31598552 -0.1230243 ]
[ 1.22865935 -2.64514773]
[-1.08888293  0.61035833]
[-0.33632435  0.23237462]
[-0.88486335  0.37580306]
[ 0.80453606 -0.78224401]
[-0.71066768 -0.98617812]
[0.87979217 1.12202986]
[ 0.01241475 -0.12962997]
[-0.81521758  0.62271886]
[0.51648194 1.33872636]
[ 0.35968119 -0.06558374]
[-1.2869822   0.88231384]
[ 2.1674511  -4.19537813]
[0.48901917 0.52945943]
[ 0.32398831 -0.45291135]
[-0.80099642  0.28179583]
[0.8692776  0.69073504]
[ 1.02026744 -1.85144441]
[0.15579553 0.3775056 ]
[-0.71681466  0.65155038]
[ 0.58563813 -1.69906472]
[-0.64626775 -0.97340041]
[ 0.6551986  -1.57015606]
[ 1.21891457 -1.46644228]
[-1.22088978 -0.67935261]
[-0.92922543  1.94953287]
[ 0.54821773 -0.03648105]
[0.5270932  0.29302847]
[ 1.66720679 -1.87519057]
[0.11916006 0.4390565 ]
[ 0.93577915 -0.66382343]
[ 0.49964988 -0.53906699]
[-0.59706434  1.94498604]
[ 1.18979155 -1.9245096 ]
[-0.16948353  0.71278183]
[ 0.80491167 -0.62918565]
[0.57205985 0.77939219]
```

```
[-0.94022287  0.6860923 ]
[-0.2014482  -0.25593045]
[-0.12338099  1.98288033]
[-2.02817495  0.60399652]
[1.72906444 0.86998928]
[-0.02646787 -1.0750443 ]
[-0.63435589  2.03743667]
[-0.92332876  0.4589538 ]
[-0.53660842  1.93553599]
[-0.57884421  0.55621257]
[ 0.87493315 -0.8537629 ]
[-0.08053237 -0.62776851]
[-1.64576437 -0.16248091]
[ 0.80846611 -0.74032796]
[-1.38824949  2.06870573]
[-0.96910891  0.5104202 ]
[0.24047363 0.18515674]
[ 2.60076189 -0.48666521]
[ 0.42736281 -0.89750232]
[ 0.47386151 -1.33068084]
[-0.90503682  0.30002713]
[ 0.81465688 -0.52559101]
[-0.39723526 -0.0360226 ]
[ 1.60847878 -1.53718772]
[0.06627732 2.04954404]
[-2.00716881  0.2258616 ]
[-0.01560438 -0.03297139]
[-0.67686058  0.0702621 ]
[0.76121119 0.77279541]
[-0.78709681 -0.15472608]
[1.02624504 1.54739465]
[ 1.45724398 -2.23800095]
[-0.32100419 -1.0439944 ]
[-0.78781662  0.51291542]
[-1.98597108  0.20524426]
[-1.57191428  2.53260893]
[-2.0657652   1.33482205]
[-0.1139418   1.91233346]
[ 0.27374684 -0.72674762]
[-0.22802708  1.74625469]
[0.52437035 0.93185151]
[-0.72296151 -0.99871583]
[0.12383625 1.52823668]
[1.16116132 1.21136541]
[ 0.59795903 -1.8533481 ]
[-0.39341476  1.35042279]
[-0.43424925 -0.22939725]
[ 1.57038009 -1.75662105]
[1.27911386 0.86502813]
[-0.89428774  1.07629824]
[-0.81391944 -1.21129968]
[-0.7978185  -0.06349326]
[-2.39598082  0.52068667]
[ 0.94356956 -1.79029752]
[ 0.56631478 -1.14242228]
```

```
[ 0.38679548 -0.5231466 ]
[-0.82060843 -0.83309907]
[-0.88534231  0.4368841 ]
[-0.72270347 -0.95007799]
[-0.97230256 -0.02680919]
[ 0.95623256 -2.18357107]
[-0.28609824 -0.74112728]
[ 1.1931934  -1.85765413]
[ 0.63407792 -1.65770159]
[-1.5752167  -0.45157034]
[1.7662413  0.80188895]
[1.95452925 0.7642341 ]
[ 0.06378791 -0.12778827]
[1.78851494 0.70454766]
[0.62656447 1.6920413 ]
[-0.499173    0.19411445]
[-1.20760712  0.32799536]
[-1.13899884  0.54782147]
[-0.32118156  0.00568035]
[-0.89137044  0.80455617]
[ 0.43372322 -0.25540054]
[ 0.66378493 -1.71231469]
[0.9744696  1.24006585]
[-0.14234692 -1.94788927]
[ 0.58671558 -1.5091447 ]
[-1.27885252  1.13727193]
[-0.79696497  0.07604189]
[ 1.47574627 -2.02908613]
[0.99206084 1.62441782]
[ 1.44638432 -2.95448702]
[ 0.80059374 -0.94764208]
[ 1.37079352 -1.27071735]
[-1.51416006  0.04784968]
[0.24222145 0.93583714]
[-2.15969336  0.73978024]
[ 1.63518159 -0.39720212]
[-0.22197967 -2.05231586]
[-0.81448665  0.34664899]
[-0.62672953 -0.27650695]
[2.12573473 0.53448874]
[1.23045514 1.19046721]
[-1.07150398 -0.67844808]
[0.58757716 2.26425843]
[ 0.07242175 -0.15905497]
[-1.44044978  1.51021198]
[0.25393874 0.16622184]
[-0.64461396  2.04265527]
[-0.26463802 -0.11698675]
[-1.05211142  0.43854887]
[-0.53277025 -0.14679433]
[0.26387781 0.94712458]
[-0.82209411 -0.48698706]
[-0.83492243  0.27583329]
[1.28753172 1.01178124]
[ 0.40703569 -0.16998365]
```

```
[0.54748788 0.56985071]
[ 0.6545547 -1.4277782]
[-1.66000761 -0.36910095]
[-0.72292562  0.40733755]
[-2.09598063  0.41533474]
[ 1.45052574 -1.59583434]
[ 0.78777524 -0.01879397]
[ 0.75578021 -1.3831111 ]
[ 1.93047125 -2.19223171]
```

Now that we have applied PCA and obtained the reduced feature set, we will check the co-relation between various Principal Components, again by using a heatmap.

In [2]: