Statistical learning and data analysis - Exercise 2 (PCA)

```
from PIL import Image
import pandas as pd
import os
import numpy as np
import matplotlib.pyplot as plt
import ssl
import statsmodels.api as sm
from scipy.sparse import csr_matrix
import seaborn as sns
import pickle
import warnings
import sys

warnings.filterwarnings("ignore")
sys.stderr = sys.stdout
plt.rcParams['figure.figsize'] = (20,10)
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Question 1

PCA on an image

In this question we evaluate the capabilities of PCA in describing information and how it may be useful in shrinking information by lowering the dimension.

A

First, we take a look at the original image and it's dimensions

---

```
im_path = '/content/drive/MyDrive/Cliff_beach_BOEM_gov.jpg'
im = Image.open(im_path)
im_array = np.array(im)
size = im_array.shape

plt.imshow(im)
plt.axis("OFF")
plt.show()
print(f"\nThe image dimensions are: {size[0]} X {size[1]}")
```

```
The image dimensions are: 1830 X 2746
```

The dimensions is the image size in pixels.

Image matrix can be described as "height x width", where height refers to the number of rows and width refers to the number of columns.

We can say that this is the size of the matrix containing the pixels of the image.

Therefore, it is a numerical representation of the amount of information stored in an image.

B

Summarize the image on it's color dimensions and perform a normalization

---

```
image_summary = np.sum(im_array, axis=2) # Summarize on the color dimensions

image_summary_normalized = (image_summary - np.mean(image_summary)) / np.std(image_summary) # Normalize
normalize_size = image_summary_normalized.shape

plt.imshow(image_summary_normalized) # plot the image
plt.axis("OFF")
plt.show()
print(f"\nThe Normalized image dimensions are: {normalize_size[0]} X {normalize_size[1]}")
```

```
The Normalized image dimensions are: 1830 X 2746
```

The quality of the image seems to remain the same, as the dimension size.

But, summarizing on the color dimensions reduced the color information in the image to a smaller number of values, therefore reducing the color depth.

C

Perform singular value decomposition (SVD) to the normalized image

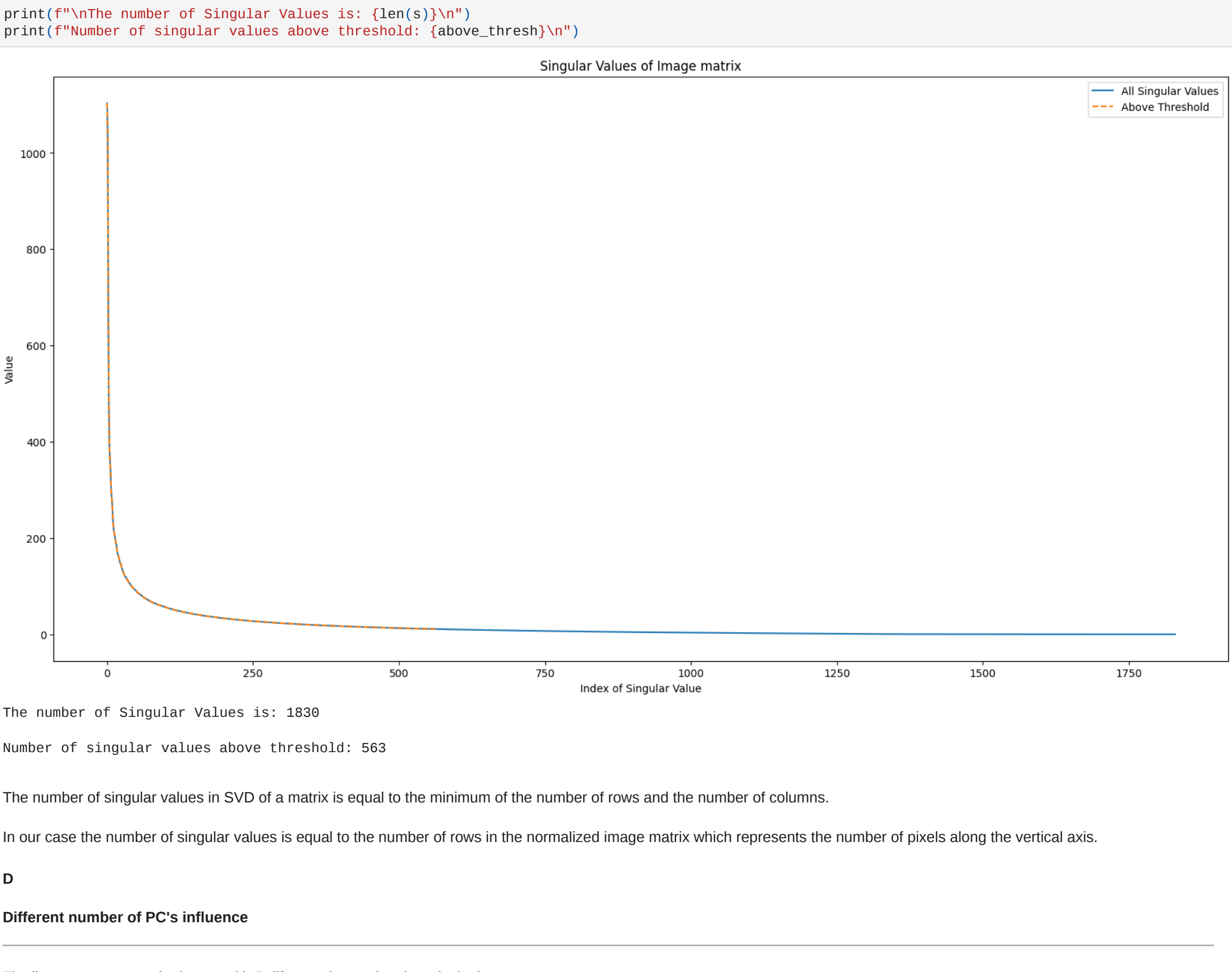
---

```
U, s, Vt = np.linalg.svd(image_summary_normalized) # SVD on image matrix

threshold = 0.01 * s.max()
relv_s = s[s > threshold]
above_thresh = np.sum(s > threshold)

plt.plot(s)
plt.plot(relv_s , '--')
plt.title('Singular Values of Image matrix')
plt.xlabel('Index of Singular Value')
plt.ylabel('Value')
plt.legend(["All Singular Values", "Above Threshold"])
plt.show()

print(f"\nThe number of Singular Values is: {len(s)}\n")
print(f"Number of singular values above threshold: {above_thresh}\n")
```




```
The number of Singular Values is: 1830

Number of singular values above threshold: 563
```

The number of singular values in SVD of a matrix is equal to the minimum of the number of rows and the number of columns.

In our case the number of singular values is equal to the number of rows in the normalized image matrix which represents the number of pixels along the vertical axis.

D

Different number of PC's influence

Finally, we recunsrtuct the image with 5 diferent sizes using the principal components.

12 principal components then 24, 48, 96, 563 (above 1% of max singular value).

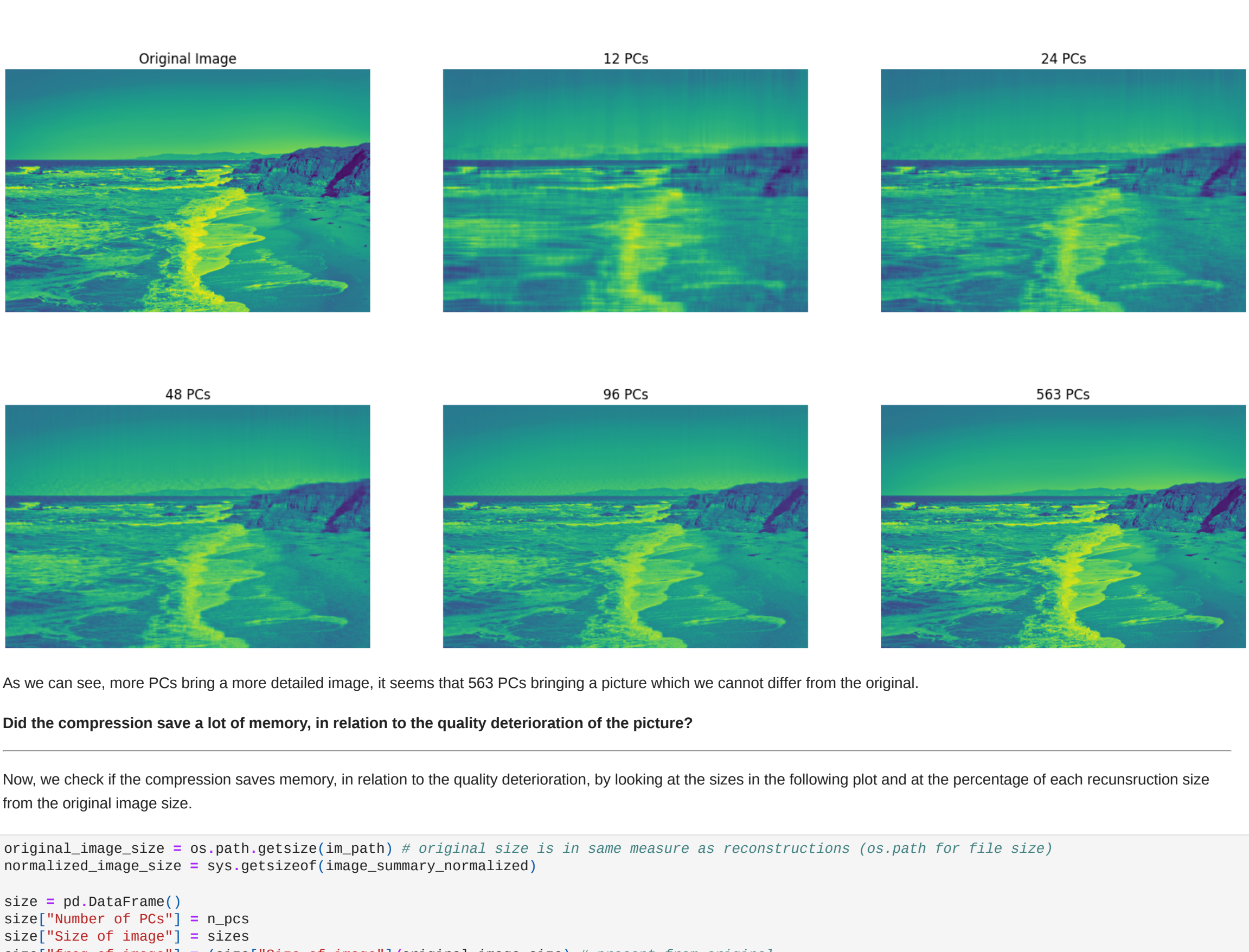
Then, we compare the image quality and sizes of the recunsructions to check the trade-off between size & quality when lowering dimension on an image matrix.

```
n_pcs = [12, 24, 48, 96, above_thresh]

fig, axs = plt.subplots(2, 3, figsize=(20, 10))

axs[0, 0].imshow(image_summary_normalized) # Original image in the first subplot
axs[0, 0].set_title('Original Image')
axs[0, 0].set_axis_off()

sizes = []
for i, n in enumerate(n_pcs):
    img_recon = np.matmul(np.matmul(U[:, :n], np.diag(s[:n])), Vt[:n, :]) # Reconstruct
    img_recon_norm = (img_recon - np.mean(img_recon)) / np.std(img_recon) # Normalize
    filename = f"recon_{n}_pcs.png"
    Image.fromarray(img_recon_norm.astype('uint8')).save(filename, format='PNG', compress_level=9)
    filesize = os.path.getsize(filename) # Get file size of reconstructed image
    sizes.append(filesize)
    axs[(i + 1) // 3, (i + 1) % 3].imshow(img_recon_norm)
    axs[(i + 1) // 3, (i + 1) % 3].set_title(f'{n} PCs')
    axs[(i + 1) // 3, (i + 1) % 3].set_axis_off()
fig.suptitle("Original Image and 5 Different Reconstructions, using PCA", fontsize=15)
plt.show()
```



As we can see, more PCs bring a more detailed image, it seems that 563 PCs bringing a picture which we cannot differ from the original.

Did the compression save a lot of memory, in relation to the quality deterioration of the picture?

Now, we check if the compression saves memory, in relation to the quality deterioration, by looking at the sizes in the following plot and at the percentage of each recunsruction size from the original image size.

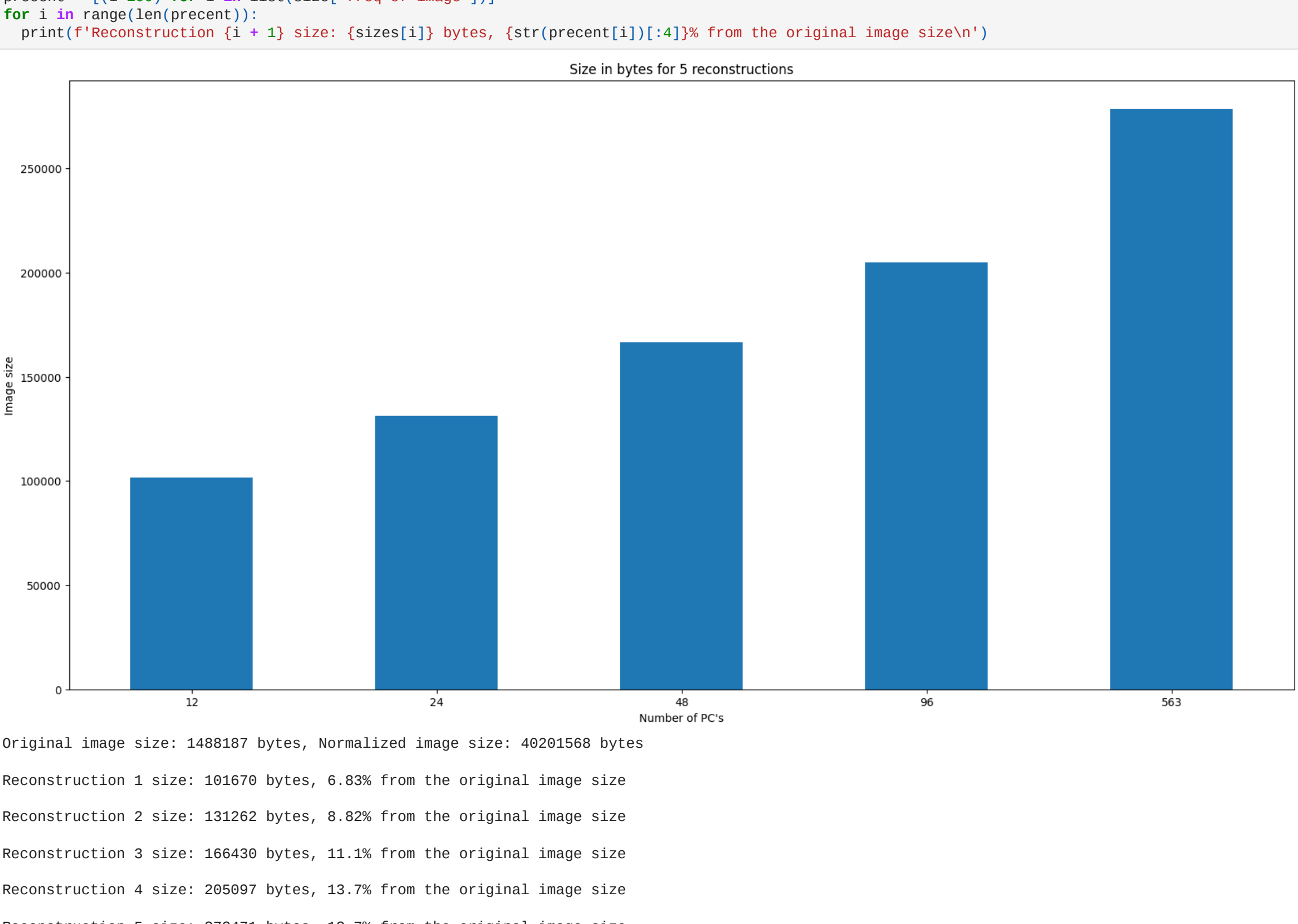
```
original_image_size = os.path.getsize(im_path) # original size is in same measure as reconstructions (os.path for file size)
normalized_image_size = sys.getsizeof(image_summary_normalized)

size = pd.DataFrame()
size["Number of PCs"] = n_pcs
size["Size of image"] = sizes
size["freq of image"] = (size["Size of image"]/original_image_size) # precent from original

size["Size of image"].plot.bar()
plt.xticks(size.index, size["Number of PCs"], rotation = 360)
plt.xlabel("Number of PC's")
plt.ylabel("Image size")
plt.title("Size in bytes for 5 reconstructions")
plt.show()


print(f'\nOriginal image size: {original_image_size} bytes, Normalized image size: {normalized_image_size} bytes\n')

precent = [(i*100) for i in list(size["freq of image"])]
for i in range(len(precent)):
  print(f'Reconstruction {i + 1} size: {sizes[i]} bytes, {str(precent[i])[:4]}% from the original image size\n')
```




```
Original image size: 1488187 bytes, Normalized image size: 40201568 bytes

Reconstruction 1 size: 101670 bytes, 6.83% from the original image size

Reconstruction 2 size: 131262 bytes, 8.82% from the original image size

Reconstruction 3 size: 166430 bytes, 11.1% from the original image size

Reconstruction 4 size: 205097 bytes, 13.7% from the original image size

Reconstruction 5 size: 278471 bytes, 18.7% from the original image size
```

As we can see, with 563 PCs we decreased the image size by more then 80%, while the first Reconstructions lower the size by more the 90%.

However, the tradeoff in size and image quality seems to be optimal in the last Reconstruction, which is in a quality that is very similar to the original and its size is decreased by 81.3% which is a massive saving off memory.

Question 2

PCA on text

In this question we will study the history of the compilation of the Bible and how different schools of writers can be identified.

A

Load the relevent Data and check dimensions

```
f = open("/content/drive/MyDrive/exodus_pickled","rb")
data = pickle.load(f)

features = np.loadtxt('/content/drive/MyDrive/features.txt', dtype ='str', delimiter ='\t')
labels = np.loadtxt('/content/drive/MyDrive/labels.txt')

print(f"\nThe dimension of exodus_pickle is {data.shape[0]} X {data.shape[1]}")
```

```
The dimension of exodus_pickle is 1201 X 15373
```

The columns dimension in exodus_pickle reffer to features vector (15373 sets of 3 words)

While the rows dimensions (1201) reffer to the verses (each row is a vector representing a verse).

The value is postive if the feature (set of 3 words) is in the verse and 0 if not.

The labels has the same dimension as the rows of exodus_pickle (1201) as the number of verses.

Labels contain information about whether a given verse was written in priestly or non-priestly (group 1/0) and we would like to identify it according to the fetures patterns.

It's represention is a binary vector, for whether the text is from group 1 or group 0. This could be the reference variable we would like to explain.

B

Can we make a plot of the verses in their current format and check whether the groups shows a real difference between them? why?

No, that is due to the high-dimensional vector space in which the verses exist, it is not feasible to directly visualize a plot that accurately displays the contrast between groups of verses in that high dimension.

Therefore, the current format does not allow for proper differentiation to groups and analysis.

However, by using PCA to reduce the dimension with a smaller set of the first PCs, we can create a more informative plot that effectively displays the differences between groups and improves visual representation.

C

Check the dimensions of U, s, and Vt

```
U, s, Vt = np.linalg.svd(data)

print(f"\nU dimension: {U.shape[0]} X {U.shape[1]}\n")
print(f"s dimension: {s.shape[0]} X 1\n")
print(f"Vt dimension: {Vt.shape[0]} X {Vt.shape[1]}\n")
```

```
U dimension: 1201 X 1201

s dimension: 1201 X 1

Vt dimension: 15373 X 15373
```

To determine which component expresses the importance vectors of each feature to each PC, we need to identify the component which its dimension corresponds to the number of features.

In this case, since the rows and columns of Vt correspond to the features, we can tell that the component that expresses the importance vectors of each feature to each PC is Vt, which has dimensions of 15373 X 15373, while 15373 is the number of features.

D

Top 10 features, with highest absolute value, for the first PC

```
pc_1 = Vt[0, : ]

top_10_pc_1 = pd.DataFrame(abs(pc_1)).T.sort_values(0, ascending = False)[:10]

print("\nTop 10 Features:\n")
for index, feature in enumerate(features[top_10_pc_1.index]):
  print(f"{index + 1}. {feature}\n")
```

```
Top 10 Features:

1. ('ו', 'עשה', 'את')

2. ('עשה', 'את', 'ה')

3. ('ו', 'צפה', 'את')

4. ('ו', 'את', 'ה')

5. ('אמה', 'ו', 'חצי')

6. ('ו', 'נתן', 'את')

7. ('ה', 'אחד', 'ו')

8. ('צפה', 'את', 'זהב')

9. ('ו', 'עשה', 'ל')

10. ('תכלת', 'ו', 'ארגמן')
```

Importance of those features

The importance of the top ten features for the first PC suggests that they are the most influential in explaining the variation captured by the first PC.

Higher absolute value of the loading for a feature suggest more contribution of the feature to the variation captured by the PC.

Therefore, the top ten features with the highest absolute values are the most important features for the first PC.

By examining these features, we can gain insight into what characteristics of the data are most strongly associated with the first PC.

what is common to most of them?

The common to most of them is the letter 'ו' and the word 'את', most of them describing doing an action: 'עשה את','צפה את','נתן את'

E

The importance of those words to this PC

As we described, those features alone explain a big part of the first PC variance, By analyse them, we can identify which patterns of the data are most strongly correlated with the first PC.

Moreover, if a set of words is contributing most to the first principal component, it indicates that the set is repeating frequently in the text.

This is because the first principal component represents the direction in the feature space where the data varies the most, and the sets of words that have high weights in the first principal component are the ones that contribute the most to the overall variation in the data.

what we can infer from this words about the text

We might assume that the origin of those words is chapter 25 of exodus. Those are sets of words that by looking at them we can infer the charasterstic of this chapter.

For example, according to the top 10 features for this chapter text we might say that it is a story about building something and covering it with gold and blue and crimson, we can infer that the size of this is 'אמה וחצי' all of this only by looking at those top 10 sets of words without even reading the chapter.

After reading the chapter we can say that it is indeed about the instructions given by god to moses on how to build the Tabernacle.

F

Project each line onto the first two principal components

As we know, the SVD of a data matrix X can be written as X = U *S* V^T.

U and V are are orthogonal matrices, and S is a diagonal matrix containing the singular values of X.

The first k columns of U and V will represent the k-dimensional linear subspace spanned by the first k principal components.

Then, we can project the data onto the 2 first principal components to get a new representation of the data in a 2-dimensional linear subspace.

To do this, we take the data matrix and multiply it with the first two rows of V using matrix multiplication, this gives us a matrix of new data points in a two-dimensional space (1201 X 2).

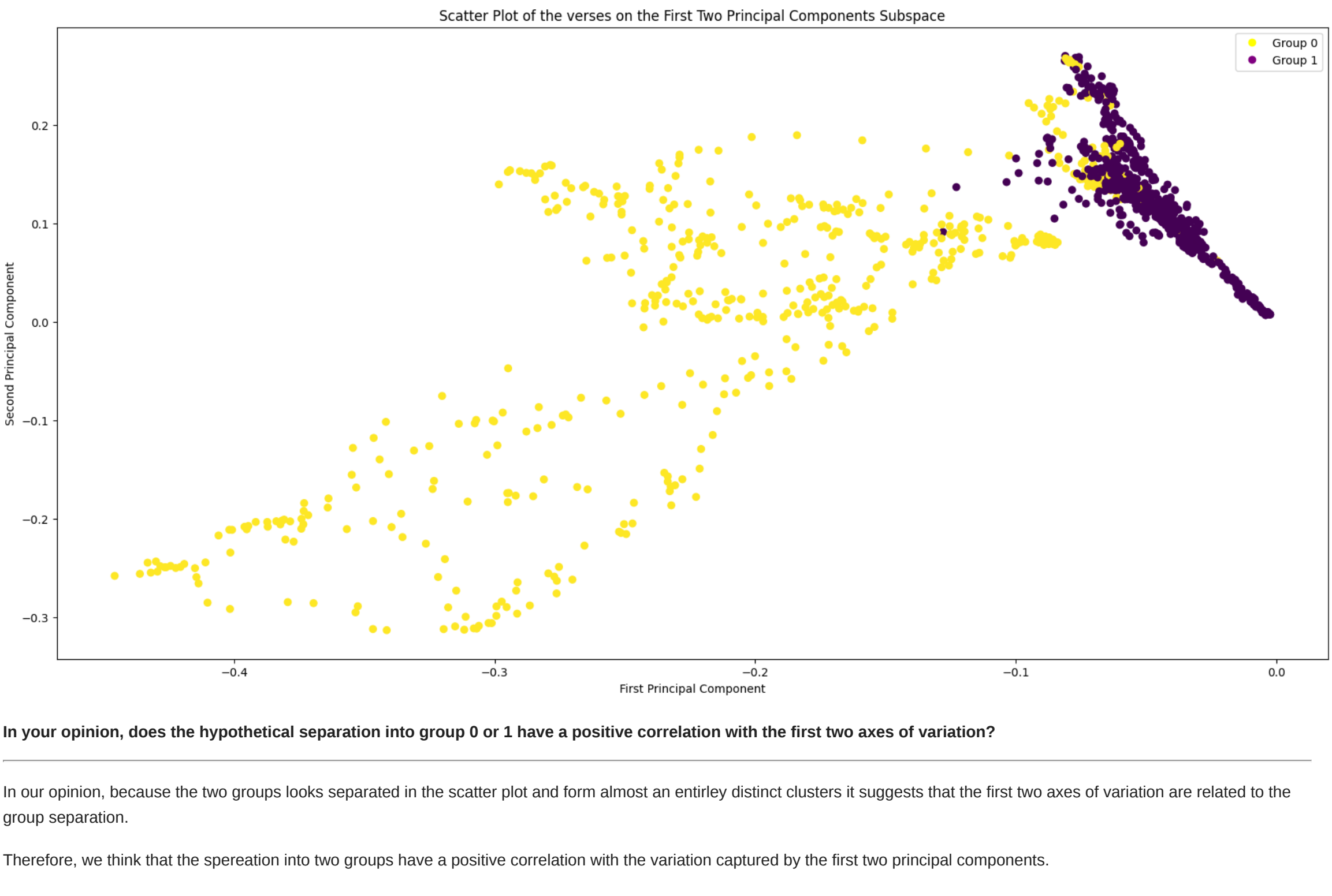
```
pc_1_2_proj = np.dot(data, Vt[:2,:].T) # relvent matrix multiplication
print(f"\nThe new subspace dimension is: {pc_1_2_proj.shape[0]} X {pc_1_2_proj.shape[1]}, As we expected\n")
```

```
The new subspace dimension is: 1201 X 2, As we expected
```

Now, we can plot on a scatter plot the 2 dimensional subspace we created that is spanned by the first 2 principal components.

In this plot we can seperate better the groups, which we could not do before on the high dimensional space of our original data.

```
plt.scatter(np.array(pc_1_2_proj[:,0]), np.array(pc_1_2_proj[:,1]), c = labels)
plt.xlabel('First Principal Component')
plt.ylabel('Second Principal Component')
plt.title('Scatter Plot of the verses on the First Two Principal Components Subspace')
color_dict = {"Group 0": 'Yellow', "Group 1": 'Purple'}
handles = [plt.plot([],[], marker="o", ls="", color=color_dict[i], label="{}".format(i))[0] for i in color_dict.keys()]
plt.legend(handles = handles)
plt.show()
```



In your opinion, does the hypothetical separation into group 0 or 1 have a positive correlation with the first two axes of variation?

In our opinion, because the two groups looks separated in the scatter plot and form almost an entirley distinct clusters it suggests that the first two axes of variation are related to the group separation.

Therefore, we think that the spereation into two groups have a positive correlation with the variation captured by the first two principal components.

What is the meaning of the graphic result of the PCA in relation to the attempt to identify the separation between the groups 0 and 1 by counting occurrences of three words in the verses?

PCA can help identify patterns in the data that may not be obvious from examining individual features.

By projecting the data onto a lower-dimensional subspace that captures the most important sources of variation we can reveal new structures and relationships in the data.

If the two groups looks well-separated in the scatter plot we can conclude that there may be a pattern or a structure in the data that can be used to predict the group membership based on the features (occurrences of three words).

Summarize & findings - Question 2 :

The research question & Data we use:

In this question, our research question is whether we can identify different writers groups in the Book of Exodus using principal components analysis.

We worked on exodus dataset, this dataset contains all the verses in the rows and features in the columns, for a given feature column it has a postivie value if the feature is in the verse, else zero.

The features are combinations of 3 words from which we try to understand whether a verse is written by a specific group of writers (group 0/1).

If the features allow us to identify diffrenet patterns between the groups we can predict the relevent group of a verse by the features.

The labels dataset maps each verse to group 0 or group 1.

As we saw, pca can help us in seperating between the groups in lower dimensional space.

SVD contribution to the research question and new information provided by this method:

The SVD have a significant contribution to our research.

First, lowering the dimension of the data was crucial in order to build a relevent plot that can seperate the groups in some sense, which we couldn't do on the high dimensional data we started with.

Second, after lowering the dimension, we saw the critical features for the first PC, that allow us to identify patterns and be able to understand what was the whole story of chapter 25 in exodus just by reading the top 10 three words combinations that contribute the most for the explained variance, that indicates that there are repeating patterns in the data.

After plotting the 2 first principal components of the data using the SVD method and coloring by the groups we saw a well-seperated group coloring.

This indicate us that there is some pattern or a structure in the data that can be used to predict the group membership based on the features (occurrences of three words).

Meaning, the research question can be studied and answered by the use of statistical tools in order to predict the group based on the features.

other statistical tools:

Another statistical tool we know and can be a good fit to this problem is Linear Regression or a different kind of regression tools we are yet to study.

In a regression model we can look at the features as the expalining variables (X matrix) while the reference variable (Y) is the lables vector, which we would like to study based on features patterns.

we can also use principal components regression (PCR) using the selected PC with the most variance and applying a linear regression model using those components to clasify the verses to the labels.