Concept and applications of Principal Component Analysis (PCA) in the context of biomedical data analysis:

Principle Component Analysis (PCA)

Principle Component Analysis (PCA) is a tool which performs a procedure that alters the original variables of a dataset into a new format. This does not alter the dataset but rather changes how it's represented. These new variables become statistically uncorrelated, termed as the 'Principal Components'.

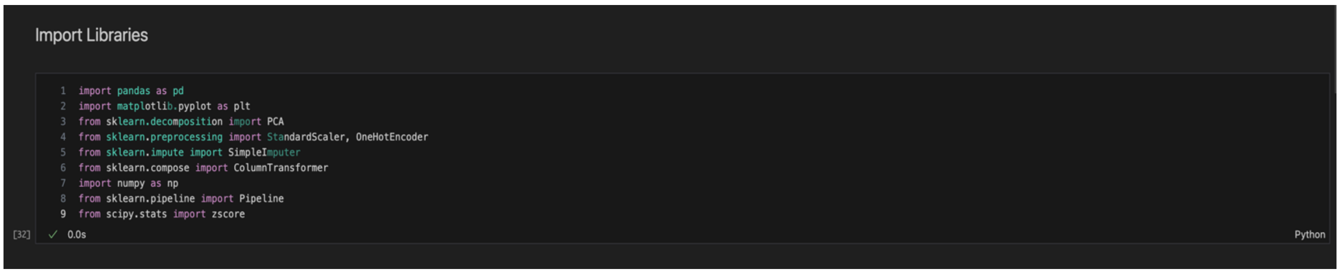
This means that the changes in one variable are not associated with changes in another. This tool is important for reducing redundancy within the dataset and ultimately makes the data easier to analyze and visualize.

These principal components are created using a linear combination of the original variables. This further facilitates PCA to reduce the number of variables within the dataset, while still attempting to preserve as much relevant information as possible.

Key features in applications of PCA are:

- Process only works with numeric features.
- PCA is sensitive to scale.
- Remove/Constraint outliers due to influence on result.

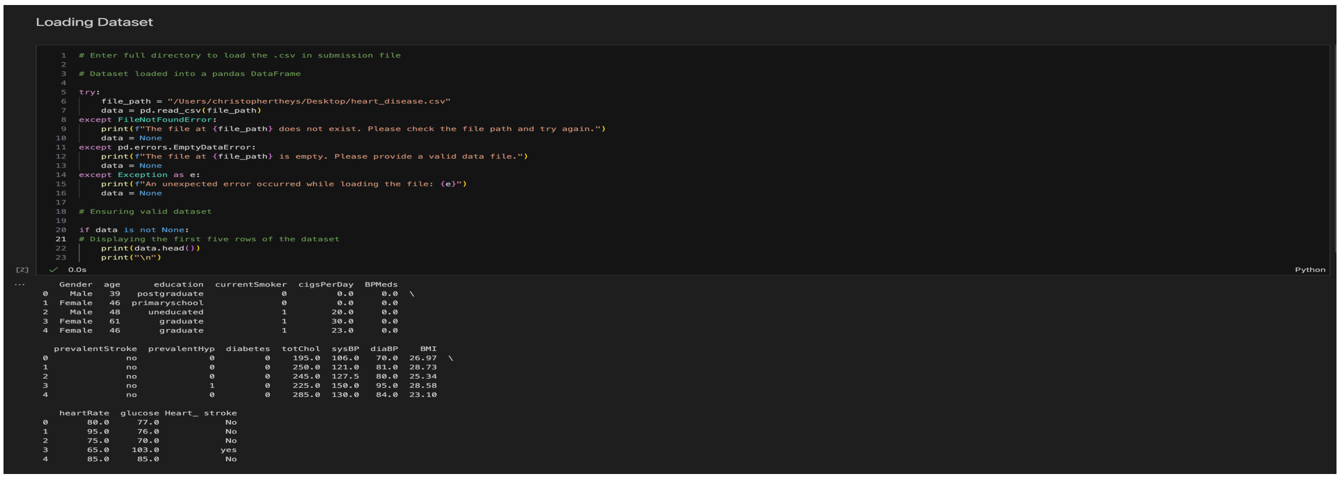
Application and Process:



Import Libraries

```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
import numpy as np
from sklearn.pipeline import Pipeline
from scipy.stats import zscore
```

[32] ✓ 0.0s Python



Loading Dataset

```python
# Enter full directory to load the .csv in submission file

# Dataset loaded into a pandas DataFrame

try:
    file_path = "/Users/christophertheys/Desktop/heart_disease.csv"
    data = pd.read_csv(file_path)
except FileNotFoundError:
    print(f"The file at {file_path} does not exist. Please check the file path and try again.")
    data = None
except pd.errors.EmptyDataError:
    print(f"The file at {file_path} is empty. Please provide a valid data file.")
    data = None
except Exception as e:
    print(f"An unexpected error occurred while loading the file: {e}")
    data = None

# Ensuring valid dataset

if data is not None:
# Displaying the first five rows of the dataset
    print(data.head())
    print("\n")
```

[2] ✓ 0.0s Python

```
   Gender  age     education  currentSmoker  cigsPerDay  BPMeds  \
0    Male   39  postgraduate              0         0.0     0.0
1  Female   46  primaryschool             0         0.0     0.0
2    Male   48    uneducated              1        20.0     0.0
3  Female   61      graduate              1        30.0     0.0
4  Female   46      graduate              1        23.0     0.0

  prevalentStroke  prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  \
0              no             0         0    195.0  106.0   70.0  26.97
1              no             0         0    250.0  121.0   81.0  28.73
2              no             0         0    245.0  127.5   80.0  25.34
3              no             1         0    225.0  150.0   95.0  28.58
4              no             0         0    285.0  130.0   84.0  23.10

   heartRate  glucose Heart_ stroke
0       80.0     77.0            No
1       95.0     76.0            No
2       75.0     70.0            No
3       65.0    103.0           yes
4       85.0     85.0            No
```

Preprocessing insights

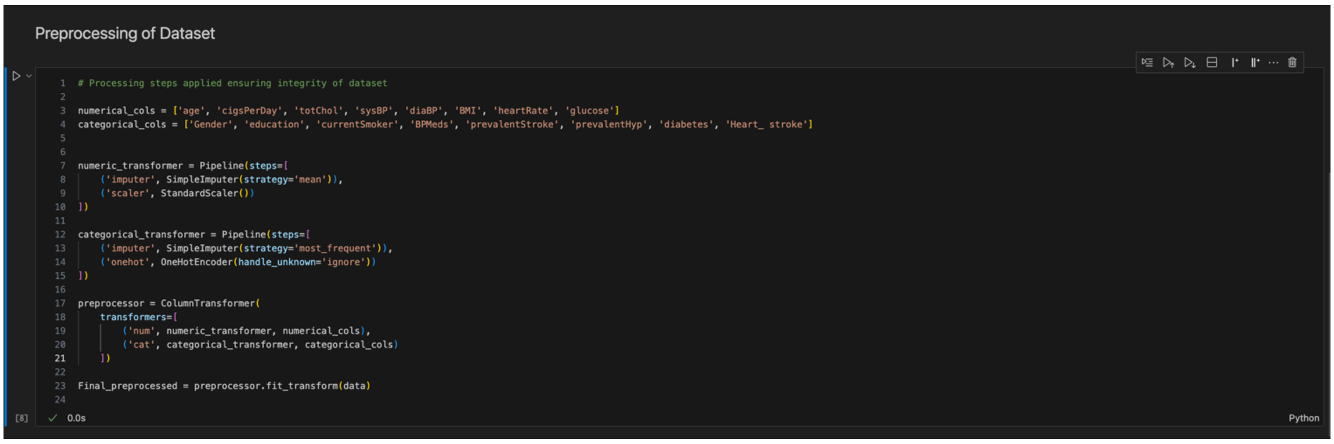
Through analyzing the dataset, these feature types were identified:

- Numerical - num_features
- Categorical - cat_features

Numerical feature preprocessing –

```
numeric_transformer = Pipeline(steps=[
    ('Imputer', SimpleImputer(strategy='mean'))
```

Snippet indicates the method used in handling missing values for numerical features. The choice of 'mean' was the best suitable strategy for this assessment. This will indicate missing values and then be replaced by the mean value of each respective feature/column to aim in preserving data distribution.



Preprocessing of Dataset

```
# Processing steps applied ensuring integrity of dataset

numerical_cols = ['age', 'cigsPerDay', 'totChol', 'sysBP', 'diaBP', 'BMI', 'heartRate', 'glucose']
categorical_cols = ['Gender', 'education', 'currentSmoker', 'BPMeds', 'prevalentStroke', 'prevalentHyp', 'diabetes', 'Heart_ stroke']


numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numerical_cols),
        ('cat', categorical_transformer, categorical_cols)
    ])

Final_preprocessed = preprocessor.fit_transform(data)

```

[8] ✓ 0.0s Python

PCA Application

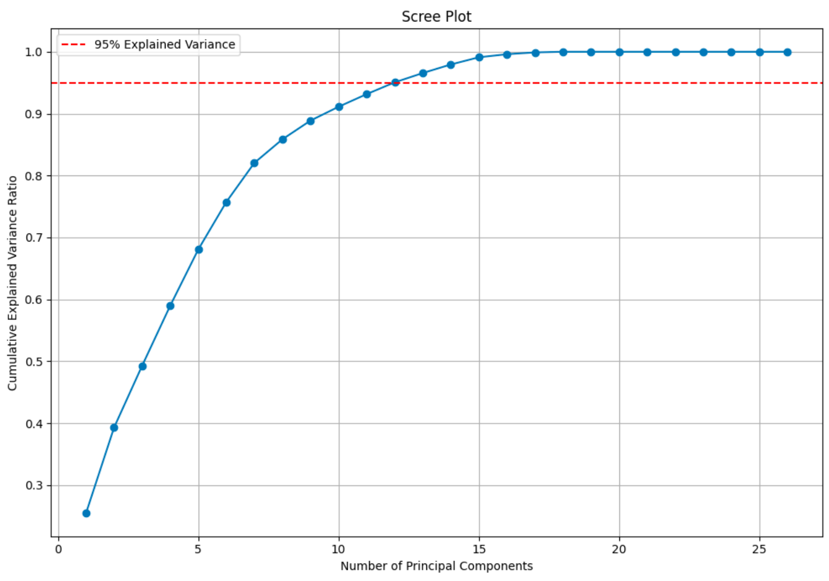
When choosing the right number of principal components, after some research I discovered that it often involves a combination of examining the explained variance and considering the specific use-case and objectives of the analysis.

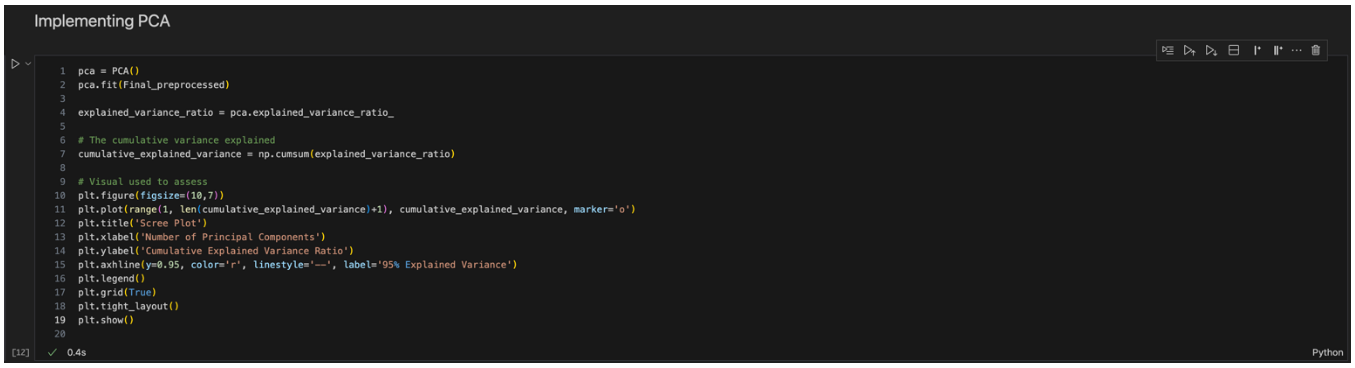
One approach, which is most common to take, is looking at the cumulative explained variance ratio as a function of the number of components. A visualization is created to assist in its determination. Plotting visuals utilized to aim in deciding the number of components.

This visual aids in visually assessing how many components are necessary to explain enough variance.

Analysis of results obtained from PCA:

Implications of the extracted principal components

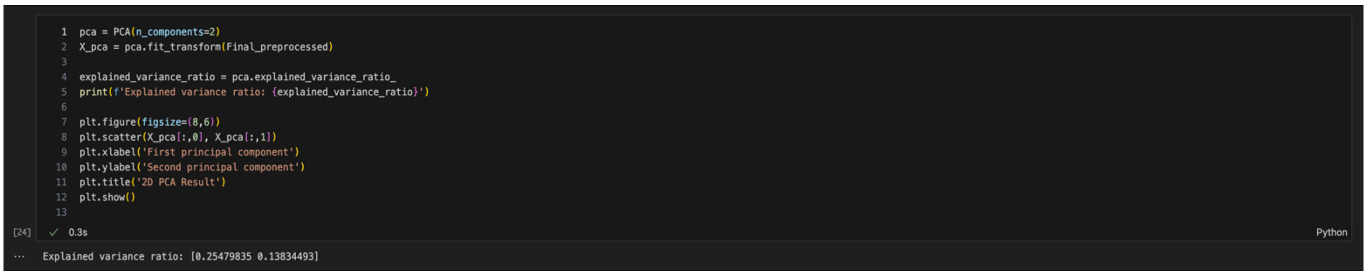



Implementing PCA

```python
pca = PCA()
pca.fit(Final_preprocessed)

explained_variance_ratio = pca.explained_variance_ratio_

# The cumulative variance explained
cumulative_explained_variance = np.cumsum(explained_variance_ratio)

# Visual used to assess
plt.figure(figsize=(10,7))
plt.plot(range(1, len(cumulative_explained_variance)+1), cumulative_explained_variance, marker='o')
plt.title('Scree Plot')
plt.xlabel('Number of Principal Components')
plt.ylabel('Cumulative Explained Variance Ratio')
plt.axhline(y=0.95, color='r', linestyle='--', label='95% Explained Variance')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

[12] ✓ 0.4s Python

Application of using 2 for the number of principle components:



```python
pca = PCA(n_components=2)
X_pca = pca.fit_transform(Final_preprocessed)

explained_variance_ratio = pca.explained_variance_ratio_
print(f'Explained variance ratio: {explained_variance_ratio}')

plt.figure(figsize=(8,6))
plt.scatter(X_pca[:,0], X_pca[:,1])
plt.xlabel('First principal component')
plt.ylabel('Second principal component')
plt.title('2D PCA Result')
plt.show()
```

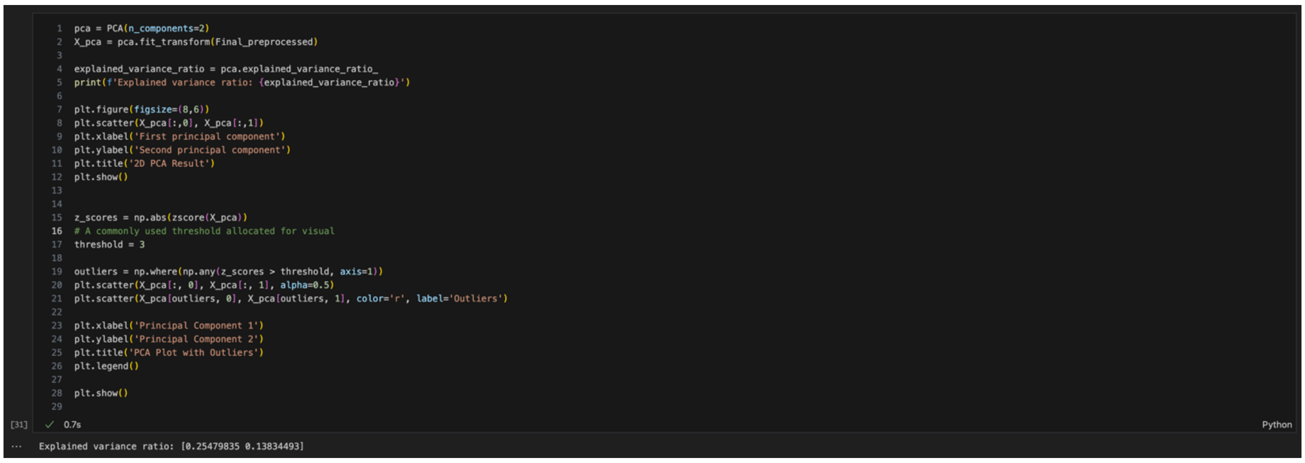
[24] ✓ 0.3s Python

```
Explained variance ratio: [0.25479835 0.13834493]
```

- 25.48% - The first principal component accounts for approximately 25.48% of the total variance in the dataset. This is substantial (a fourth of the total dataset) considering that this is a reduction from the original feature space from the dataset.

- 13.83% - The second principal component accounts for approximately 13.83% of the total variance in the dataset. The additional 13.83% brings the cumulative retained variance to 39.31%

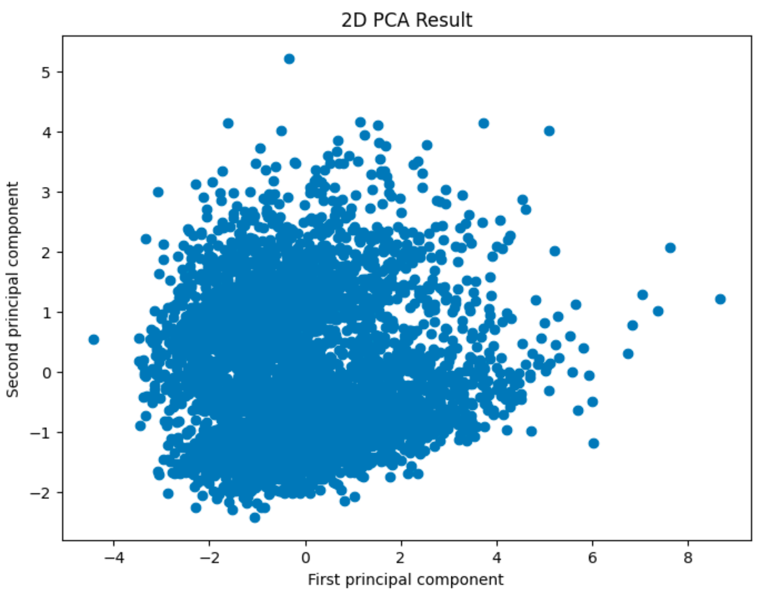
Additional code of visualizing results

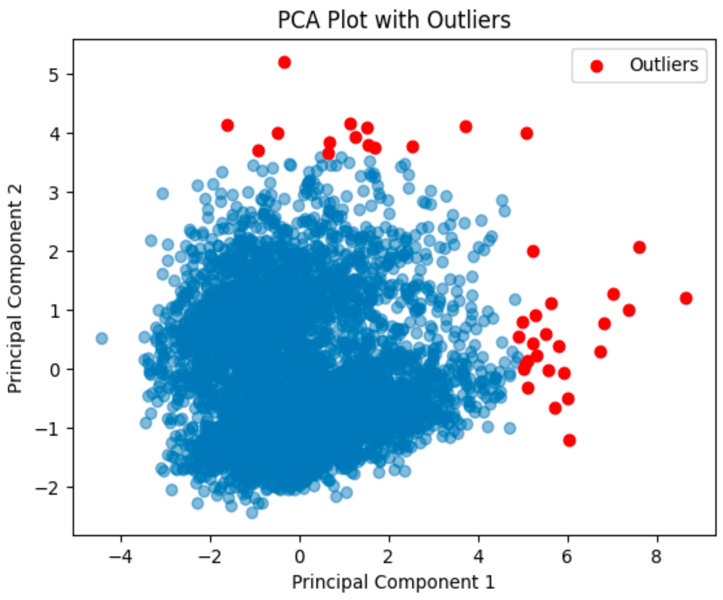


```python
pca = PCA(n_components=2)
X_pca = pca.fit_transform(Final_preprocessed)

explained_variance_ratio = pca.explained_variance_ratio_
print(f'Explained variance ratio: {explained_variance_ratio}')

plt.figure(figsize=(8,6))
plt.scatter(X_pca[:,0], X_pca[:,1])
plt.xlabel('First principal component')
plt.ylabel('Second principal component')
plt.title('2D PCA Result')
plt.show()


z_scores = np.abs(zscore(X_pca))
# A commonly used threshold allocated for visual
threshold = 3

outliers = np.where(np.any(z_scores > threshold, axis=1))
plt.scatter(X_pca[:, 0], X_pca[:, 1], alpha=0.5)
plt.scatter(X_pca[outliers, 0], X_pca[outliers, 1], color='r', label='Outliers')

plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA Plot with Outliers')
plt.legend()

plt.show()

```

```
[31]  ✓ 0.7s                                                                Python
...   Explained variance ratio: [0.25479835 0.13834493]
```

Visualization of results:





Insights in process:

Explanations

Dataset – Heart Disease Dataset
Source - https://www.kaggle.com/code/ekrembayar/heart-disease-uci-eda-pca-kmeans-hc-rf-with-r

Few issues initially faced was identifying a suitable dataset for meeting the objective of Principle Component Analysis (PCA).
General elements which are vital in achieving the objective of PCA are:

- Dimensionality Reduction
- Visualization
- Noise Reduction
- Feature Engineering
- Interpretation of Variability

Understanding the purpose of PCA was essential in selecting a dataset which meets key requirements for suitable application of the concept.
Similarly, the dataset selected should encompass these vital elements:

- Numerical Data
- High Dimensionality
- There are correlated variables.
- Has sufficient sample size.
- Able to handle preprocessing.
- Evidence of linearity

Reference List:

- Jonathon Shlens, (2014), Version 3.02, '*A Tutorial on Principal Component Analysis'.* https://arxiv.org/pdf/1404.1100&sa=U&ved=2ahUKEwi57Mfr0ZDpAhWtF6YKHfSxAck4ChAWMAZ6BAgEEAE&usg=AOvVaw2ccduDFnmcXvF-iGE-VXIM

- Md. Touhidul Islam, Sanjida Reza Rafa, Md. Golam Kibria. (2020). *Early Prediction of Heart Disease Using PCA and Hybrid Genetic Algorithm with k-Means. https://ieeexplore.ieee.org/abstract/document/9392655?casa_token=fYksPCsZtxsAAAAA:vWAI3wrPS2gVgDCCdGl0wy9QguxBP7zmkodPC16MsbAdvrgAZU8R4M8HZyp4lIRFJJYlpDDTMM4*