PCA Stuff WorkFlow

1. Importing libraries:
    - numpy (as np): For numerical operations and arrays.
    - pandas (as pd): For data manipulation and analysis.
    - seaborn (as sns): For data visualization.
    - sklearn.metrics: For calculating evaluation metrics.
    - plotly.express (as px): For interactive visualizations.
    - matplotlib.pyplot (as plt): A library for creating static, animated, and interactive visualizations in Python.
    - Specific modules from scikit-learn (sklearn):
        - PCA: A class for performing Principal Component Analysis.
        - StandardScaler: A class for standardizing features by removing the mean and scaling to unit variance.
        - KNeighborsClassifier: A class implementing the k-nearest neighbors classifier.
        - RandomForestClassifier: A class implementing the random forest classifier.
        - LogisticRegression: A class implementing logistic regression.
        - train_test_split: A function for splitting data into training and testing sets.

2. Importing data:
    - Reads a CSV file into a pandas DataFrame (df).

3. Data exploration:
    - cols: A list of column names representing different body measurements.
    - target: A list of target labels representing different clothing sizes.
    - target_num: A list of target numbers assigned to each clothing size.
    - Prints the first few rows of the DataFrame using df.head().
    - Computes descriptive statistics of the DataFrame using df.describe().

4. Data preprocessing:
    - Scales the data using StandardScaler() from sklearn.preprocessing.
    - Fits the scaler on the DataFrame (df) using scaler.fit(df).
    - Transforms the data to obtain scaled features using scaler.transform(df) and stores it in scaled_data.

5. Data visualization:
    - Creates a scatter plot using the first two columns of scaled_data (scaled_data[:, 0] and scaled_data[:, 1]) as the x and y coordinates, respectively.
    - Colors the points based on the corresponding target values (target_num).
    - Displays the plot using plt.show().

6. Data splitting:
    - Splits the scaled data (scaled_data) and target values (target_num) into training and testing sets using train_test_split() from sklearn.model_selection.
    - The training set consists of 80% of the data, and the remaining 20% is used for testing.
    - The split is performed in a stratified manner, preserving the distribution of target values.

7. Logistic Regression before PCA:
    - Creates an instance of LogisticRegression() from sklearn.linear_model.
    - Fits the logistic regression model on the training data using reg_model.fit(x_train, y_train).
    - Computes the accuracy score of the model on the test data using reg_model.score(x_test, y_test) and stores it in reg_score.
    - Makes predictions on the test data using reg_model.predict(x_test) and stores the predicted labels in reg_pred.
    - Computes the confusion matrix using metrics.confusion_matrix(y_test, reg_pred) and stores it in reg_cm.
    - Computes various evaluation metrics (accuracy, precision, recall, F1 score) using functions from metrics module.
    - Prints the evaluation metrics.

8. Random Forest before PCA:
    - Similar to logistic regression, but uses RandomForestClassifier() from sklearn.ensemble.
    - Fits the random forest model, computes the evaluation metrics, and prints them.

9. K-Nearest Neighbors (KNN) before PCA:
    - Similar to logistic regression and random forest, but uses KNeighborsClassifier() from sklearn.neighbors.
    - Fits the KNN model, computes the evaluation metrics, and prints them.

10. PCA (Principal Component Analysis):
    - The code initializes a PCA object with a variance threshold of 0.95. This means that the resulting transformed data will retain 95% of the variance in the original data and set the test size to 20% and random state to 42 for reproducibility.
    - The fit method is called on the PCA object to perform the actual PCA on the scaled_data, which is assumed to be the input dataset.
    - The transform method is used to transform the scaled_data into the new feature space defined by the principal components. The transformed data is stored in x_pca.
    - The shape of x_pca is printed to see the number of samples and the reduced dimensionality.
    - Visualize the principal components (EigenVectors) as a heatmap using sns.heatmap from the seaborn library.

11. Logistic Regression after PCA:
    - The code splits the transformed data x_pca and the target variable target_num into training and testing sets using the train_test_split function from scikit-learn.
    - A logistic regression model (LogisticRegression) is created and trained on the training data (pca_x_train and pca_y_train) using the fit method.
    - The accuracy score of the model is computed on the test data (pca_x_test and pca_y_test) using the score method.
    - The predicted values for the test data are obtained using the predict method.
    - A confusion matrix (pca_reg_cm) is computed to evaluate the performance of the logistic regression model.

- Various performance metrics such as accuracy, precision, recall, and F1 score are computed using functions from the metrics module of scikit-learn.
- The confusion matrix is visualized using a heatmap from the seaborn library.

12. Random Forest Classifier after PCA:
    - Similar to the logistic regression section, a random forest classifier (RandomForestClassifier) is created and trained on the transformed data (pca_x_train and pca_y_train).
    - The accuracy score, predicted values, confusion matrix, and performance metrics are computed and displayed similarly to the logistic regression section.

13. K-Nearest Neighbors Classifier after PCA:
    - A k-nearest neighbors classifier (KNeighborsClassifier) is created with n_neighbors=5.
    - The classifier is trained on the transformed data (pca_x_train and pca_y_train).
    - The accuracy score, predicted values, confusion matrix, and performance metrics are computed and displayed similarly to the previous sections.

14. Visualization:
    - The scatter plot of the transformed data (x_pca) is displayed using plt.scatter from Matplotlib. The colors represent different classes in the target_num variable.
    - A 3D scatter plot of the transformed data is created using the scatter_3d function from Plotly Express. The colors represent different classes.

15. PCA Components and Explained Variance:
    - Print the principal components (pca.components_) obtained from PCA.
    - Print the explained variance ratio for each principal component (pca.explained_variance_ratio_).
    - Create a DataFrame (df_) to display the principal components with their corresponding features.
    - Plot a heatmap of the principal components using sns.heatmap from the seaborn library.

16. Explained Variance and Cumulative Variance:
    - Calculate the explained variance ratios for each principal component.
    - Calculate the cumulative explained variance by taking the cumulative sum of the explained variance ratios.
    - Create DataFrames (explained_variance_df and cumulative_variance_df) to display the explained variance and cumulative variance.
    - Visualize the explained variance ratios and cumulative variance using bar plots.

17. Scatter Plot of Loadings:
    - Create a 3D scatter plot of the loadings of each feature on the principal components.
    - The loadings represent the correlation between each feature and the principal components.

18. Scatter Plot of Scores:
    - Create a 3D scatter plot of the principal components (PC1, PC2, and PC3) colored by the class labels (Class).
    - Visualize the distribution of data points in the reduced-dimensional space.