14.01.2024,

Michał Saturczak

Data Science fashion-MNIST project

Important links:

- **Kaggle**: https://www.kaggle.com/code/majkeloess/fashion-mnist-dataset-analysis-ds-winter-project
- **Github**: https://github.com/majkeloess/fashionMnist

Introduction

The primary aim of this project is to navigate through the practical aspects of data analysis by employing the Fashion-MNIST dataset a collection of Zalando's article images. The key tasks encompass:

1. **Data Summary and Discussion**: Provide a concise overview of the dataset, focusing on the most pertinent practical aspects.

2. **Dimensionality Reduction**: Apply techniques to reduce the dimensionality of the dataset, facilitating a more manageable representation.

3. **Visualization of Reduced Dataset**: Utilize visualization methods to portray the reduced dataset effectively.

4. **Clustering**: Implement clustering algorithms on the dataset and assess the results, considering the presence of classification labels.

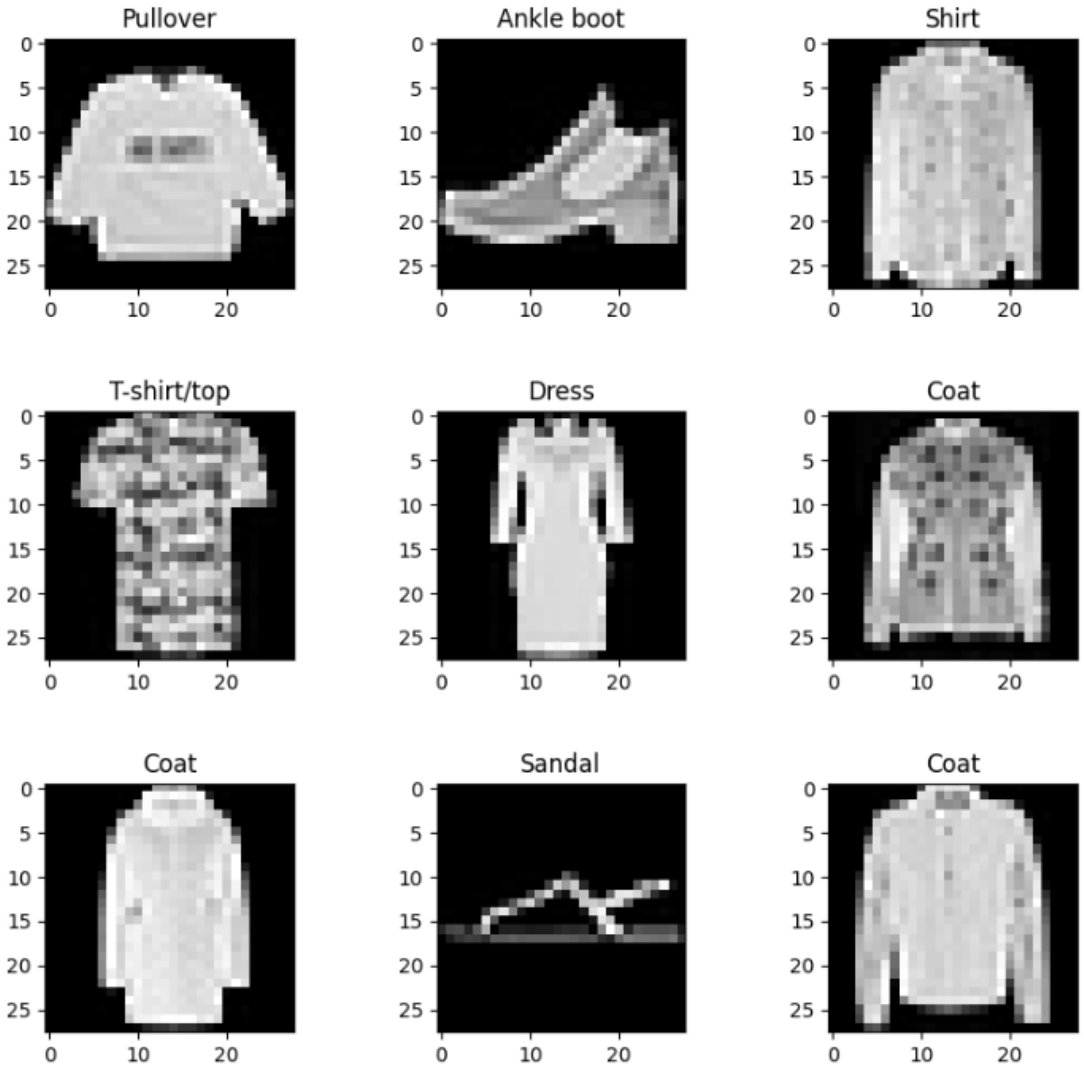
5. **Dataset Splitting**: Divide the dataset into training and testing sets, a crucial step in model development.

6. **Classification and Evaluation**: Undertake classification tasks on the dataset, followed by a rigorous evaluation of the results.

Dataset Description

The Fashion-MNIST dataset serves as the cornerstone of our exploration into practical data analysis, offering a nuanced departure from the traditional MNIST dataset. Crafted as a response to the need for more challenging validation datasets, Fashion-MNIST introduces a diverse collection of grayscale images, each representing various clothing articles from Zalando's inventory. With ten distinct classes, including T-shirts, trousers, dresses, and more, the dataset expands the horizons of image classification. Notably, the images are formatted as 28x28 grayscale representations, presenting a unique challenge and opportunity for algorithms to discern meaningful features from relatively small images.

Examples and labels:



Above, several examples of images from the dataset provide a visual glimpse into the variety of fashion items captured. The dataset representation comprises **785** columns, where the first column denotes the label of an image, and the subsequent 784 columns correspond to pixels ranging from 1 to 784. The pixel values in Fashion-MNIST range from 0 to 255, where 0 represents black and 255 represents white. The values in between 0 and 255 represent various shades of gray. Each pixel in the image is assigned a numerical value that indicates its intensity or darkness. The dataset under analysis contains a total of **60,000** records, meticulously divided into 48,000 for training and 12,000 for testing (80/20), with each record belonging to one of the ten unique labels.

Labels:

0. T-shirt/top
1. Trouser
2. Pullover
3. Dress
4. Coat
5. Sandal
6. Shirt
7. Sneaker
8. Bag
9. Ankle boot

Dimensionality Reduction

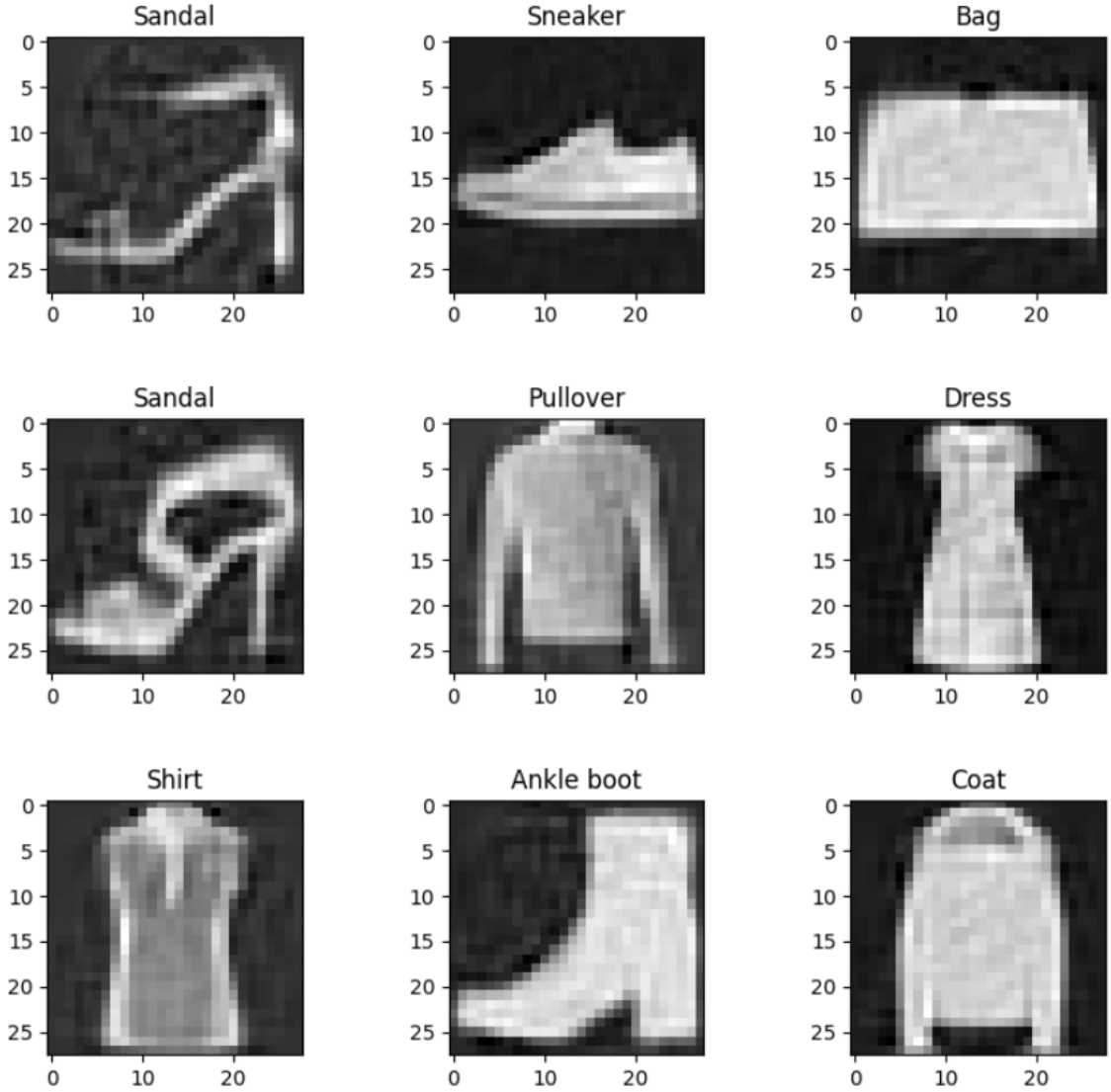
In the realm of data analysis, dimensionality reduction stands as a pivotal technique, particularly when dealing with datasets featuring a multitude of features. The essence lies in transforming high-dimensional data into a lower-dimensional representation while retaining essential information. This process not only simplifies the dataset but also aids in mitigating issues like the curse of dimensionality, improving computational efficiency, and enhancing the interpretability of results.

One prominent technique for dimensionality reduction is Principal Component Analysis (PCA). PCA identifies the principal components linear combinations of the original features capturing the maximum variance within the data. By selecting a subset of these components, we can achieve a reduced-dimensional representation that retains a significant portion of the dataset's variability.

Initially, PCA is fitted to the training data without specifying the number of components. The cumulative explained variance is then calculated, and the number of components needed to retain at least 95% of the variance is determined. Subsequently, a new PCA

object is created with the selected number of components, and the transformation is applied to both the training and test datasets.

After applying Principal Component Analysis (PCA) to reduce the dimensionality of the dataset, an intriguing aspect is the ability to reverse this transformation and visualize the reconstructed images.



This visualization provides a tangible glimpse into how the dimensionality-reduced representation impacts the appearance of the original images. It highlights the ability of PCA to capture essential features while reducing the dataset's dimensionality, enabling a more efficient representation for subsequent tasks.

Clustering using K-Means Algorithm

Clustering is a fundamental unsupervised learning technique that involves grouping similar data points into distinct clusters based on certain features. The aim is to uncover inherent patterns or structures within the data, facilitating a deeper understanding of its underlying characteristics. One popular clustering algorithm is K-Means.

K-Means is a partitioning algorithm that divides a dataset into 'k' clusters. It operates iteratively, assigning data points to clusters and updating cluster centroids until convergence. The number of clusters, 'k,' is a crucial parameter and often needs to be specified a priori.

In the context of the Fashion-MNIST dataset, K-Means clustering proves to be a suitable choice for several reasons. The dataset consists of grayscale images of fashion items, and K-Means is particularly effective when dealing with relatively well-defined and spherical clusters in the feature space. Given the nature of clothing items, such as T-shirts, trousers, and dresses, K-Means can potentially capture inherent patterns in the pixel space.

The algorithm's simplicity, scalability, and efficiency make it well-suited for large datasets like Fashion-MNIST. Additionally, the interpretability of K-Means clusters can provide insights into how different fashion items group together, potentially revealing interesting relationships and similarities. The K-Means clustering model is initialized with **10 clusters**, aligning with the number of unique labels in the dataset. This choice is made with the expectation that the clusters may align with the distinct classes of fashion items, allowing for an informative analysis of the dataset's structure.

Upon applying K-Means clustering to the Fashion-MNIST dataset, an analysis of the clustering accuracy yields a result of **0.56**. This metric is obtained by comparing the assigned cluster labels to the true labels of the dataset. A cluster analysis accuracy of 0.56 indicates that approximately **56%** of the instances were correctly assigned to their corresponding clusters.

While cluster analysis accuracy provides a quantitative measure of the algorithm's performance, it's important to interpret this result in the context of the dataset's complexity and inherent challenges. Fashion-MNIST, with its diverse array of fashion items and intricate patterns, presents a non-trivial clustering task.

The achieved accuracy of 0.56 suggests a moderate level of success in capturing inherent structures within the dataset. However, factors such as the choice of clustering algorithm, the intrinsic complexity of the dataset, and the potential overlap between fashion classes contribute to the limitations of clustering accuracy as a sole evaluation metric.

Classification using K-Nearest Neighbors

K-Nearest Neighbors (KNN) is a versatile and intuitive classification algorithm that makes predictions based on the majority class of its 'k' nearest neighbors in the feature space. KNN is particularly effective for datasets with discernible decision boundaries and instances of local patterns.

In the progression of this data science project, the inclusion of classification techniques becomes paramount. Classification allows us to go beyond clustering and assign specific labels to data instances, enabling the model to make predictions on unseen data. In the context of Fashion-MNIST, the goal is to train a classifier that can accurately categorize images of clothing items into their respective classes. KNN classifier is implemented using the reduced-dimensional data obtained after applying Principal Component Analysis (PCA) to the Fashion-MNIST dataset. This approach leverages the insights gained from dimensionality reduction to enhance the efficiency and interpretability of the classification model.

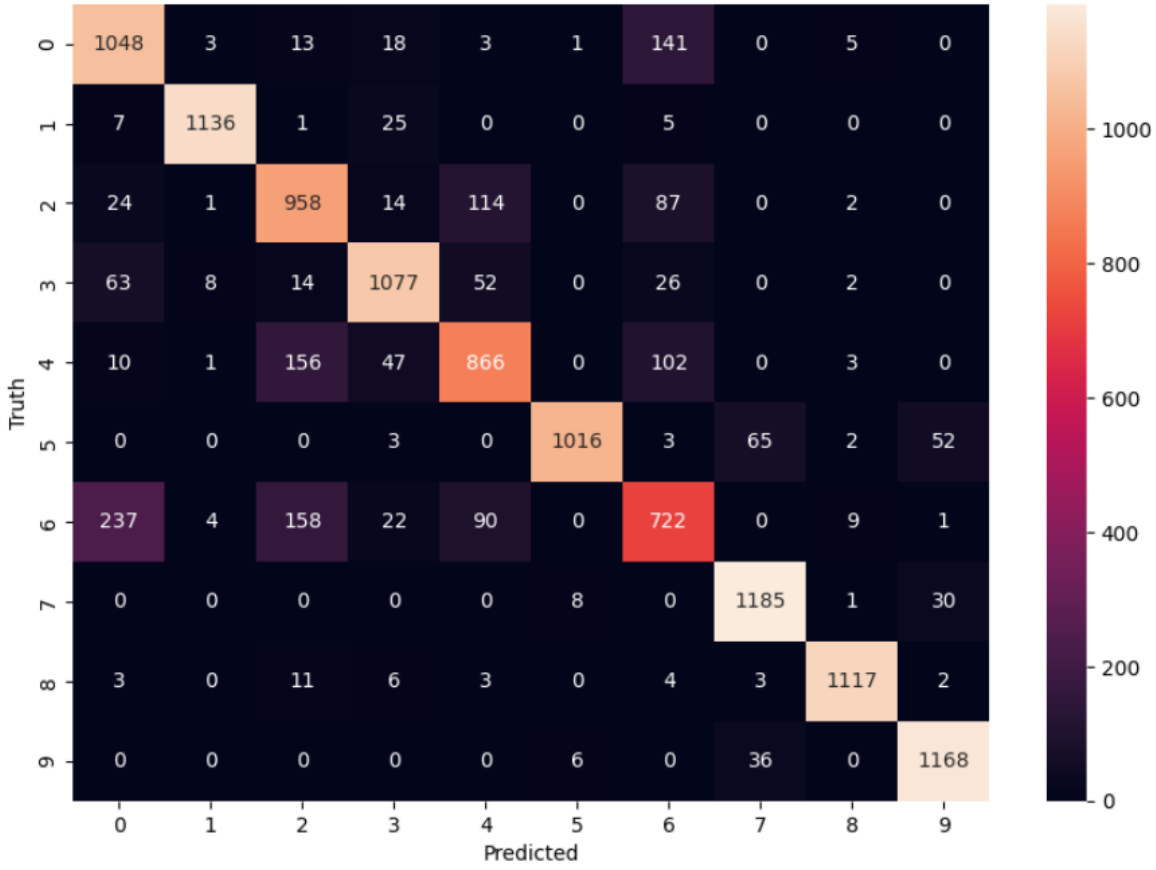
The classification report provides a detailed evaluation of the KNN classifier's performance on the Fashion-MNIST test set. The table below summarizes key metrics such as **precision**, **recall**, and **F1-score** for each class, along with overall accuracy:

	precision	*recall*	*F1-score*	*support*
0	0.75	0.85	0.80	1232
1	0.99	0.97	0.98	1174
2	0.73	0.80	0.76	1200
3	0.89	0.87	0.88	1242
4	0.77	0.73	0.75	1185
5	0.99	0.89	0.94	1141
6	0.66	0.58	0.62	1243
7	0.92	0.97	0.94	1224
8	0.98	0.97	0.98	1149
9	0.93	0.97	0.95	1210
Accuracy			0.86	12000
Macro avg	0.86	0.86	0.86	12000
Weighted avg	0.86	0.86	0.86	12000

Interpreting the Classification Report Table:

- **Precision**: The proportion of true positive predictions among all positive predictions.
- **Recall**: The proportion of true positive predictions among all actual positives.
- **F1-score**: The harmonic mean of precision and recall, providing a balance between the two metrics.
- **Support**: The number of actual occurrences of the class in the test set.

The table is organized by class (represented by numbers 0 to 9 in the Fashion-MNIST dataset). The "accuracy" row indicates the overall accuracy of the model on the entire test set. The "macro avg" row represents the average performance across all classes, while the "weighted avg" row considers the proportion of instances for each class.



The classification report provides a detailed evaluation of the KNN classifier's performance on the Fashion-MNIST test set. The **confusion matrix**, a tabular representation of true positive, true negative, false positive, and false negative

predictions, offers further insights into the classifier's behavior. The confusion matrix allows us to assess how well the classifier performs for each class, revealing patterns of misclassifications and highlighting areas for improvement. True positives, true negatives, false positives, and false negatives contribute to the overall understanding of the model's strengths and weaknesses. The heatmap visualization of the confusion matrix enhances interpretability by providing a visual representation of the classification performance. Intensity of color indicates the number of instances falling into each category.

The overall accuracy of the KNN classifier is reported to be **0.86**, indicating that approximately **86%** of the instances in the Fashion-MNIST test set were correctly classified. This metric serves as a general measure of the model's effectiveness in making accurate predictions across all classes.

While accuracy is a valuable metric, it's essential to consider its interpretation in the context of the dataset's characteristics. Fashion-MNIST, with its diverse set of clothing items, presents challenges such as variations in patterns, textures, and styles. The achieved accuracy of 0.86 suggests a substantial degree of success in capturing and categorizing these variations. Accuracy provides a high-level summary, it is recommended to delve deeper into class-specific metrics, such as precision, recall, and F1-score, to gain insights into the model's performance across individual classes. Additionally, visualizations like the confusion matrix and heatmap offer a nuanced understanding of classification patterns and potential areas for improvement.

Conclusion

In comparing the results of clustering and classification, the K-Means clustering algorithm yielded an accuracy of 0.56, indicating a moderate success in grouping similar instances within the Fashion-MNIST dataset. However, it's important to note that clustering operates in an unsupervised manner, and the clusters may not align precisely with the true class labels. On the other hand, the K-Nearest Neighbors (KNN) classifier achieved an overall accuracy of 0.86, showcasing a substantial success in categorizing clothing items based on reduced-dimensional features obtained through Principal Component Analysis (PCA). The supervised nature of classification, with explicit class labels, allows for a more direct evaluation of model performance on unseen data. The higher accuracy in classification suggests the effectiveness of supervised learning techniques in capturing the diverse patterns and styles present in the Fashion-MNIST dataset.

The project provides a foundation for understanding the practical challenges and nuances associated with real-world data analysis tasks. The success of the classification model suggests the importance of supervised learning techniques, especially in scenarios where distinct class labels are available.

In conclusion, the project journeyed through the key stages of data exploration, dimensionality reduction, clustering, and classification, highlighting the application of modern data analysis techniques to a challenging dataset like Fashion-MNIST.

References

- https://github.com/zalandoresearch/fashion-mnist
- https://www.kaggle.com/datasets/zalando-research/fashionmnist
- https://www.youtube.com/watch?v=3bHxOYczJn0
- https://www.youtube.com/watch?v=TKfTKA5EmmY&t=302s
- https://medium.com/@joel_34096/k-means-clustering-for-image-classification-a648f28bdc47
- Data Science Lectures

Comparison to ChatGPT

Your code appears well-structured and follows a clear pipeline for data loading, exploration, dimensionality reduction (PCA), clustering (KMeans), and classification (KNN). However, there are a few suggestions to enhance the code:

1. *Function Definitions:*
   - *Consider encapsulating chunks of code into functions. This can improve readability and reusability.*
   - *For example, you could create functions for loading data, performing PCA, visualizing images, and evaluating models.*

2. *Comments and Documentation:*
   - *Add comments to explain the purpose and functionality of crucial code segments. This makes the code more understandable, especially for others who might read or collaborate on the project.*
   - *Consider adding docstrings to your functions, providing information about their parameters and return values.*

3. *Visualization:*
   - *Instead of repeating visualization settings in multiple places, define them once at the beginning or in a separate function. This ensures consistency across visualizations.*
   - *Add titles to your plots for clarity.*
4. *Consistent Variable Naming:*
   - *Maintain consistent variable naming conventions throughout the code. For example, you use both snake_case (n_components_95) and camelCase (cumulative_explained_variance). Stick to one convention for better code readability.*

5. *Error Handling:*
   - *Consider adding error handling for potential issues, such as cases where the dataset file is not found or if the PCA or clustering steps encounter problems.*

6. *Modularize the Code:*
   - *Break down the code into smaller, modular functions or classes. This promotes better organization and makes it easier to understand and maintain.*

Here's a modified version incorporating some of these suggestions:

<CODE SUGGESTED BY CHAT IS INCLUDED IN REPOSITORY>

Thank you for taking the time to read this report! I hope you found it enjoyable!!

Michał Saturczak