Principal Component Analysis and Variational Autoencoder for Dimensionality Reduction and Reconstruction

---

1. Introduction

This report presents a comprehensive analysis and modeling pipeline involving Principal Component Analysis (PCA) and a Variational Autoencoder (VAE) applied to the data_processed_4039.csv dataset. The primary objectives are dimensionality reduction, feature extraction, and evaluating the reconstruction quality of data using VAE.

---

2. Dataset Description

- **File name:** data_processed_4039.csv
- **Rows:** 4039 observations
- **Features:** Multiple engineered features relating to SPREAD, lag values, rolling volatility, and scaled/standardized indicators.
- **Target variables removed:**
  - SPREAD*_diff._percentage._60*_scale._standardize
  - SPREAD*_diff._percentage._60*_scale._standardize*_categorized

---

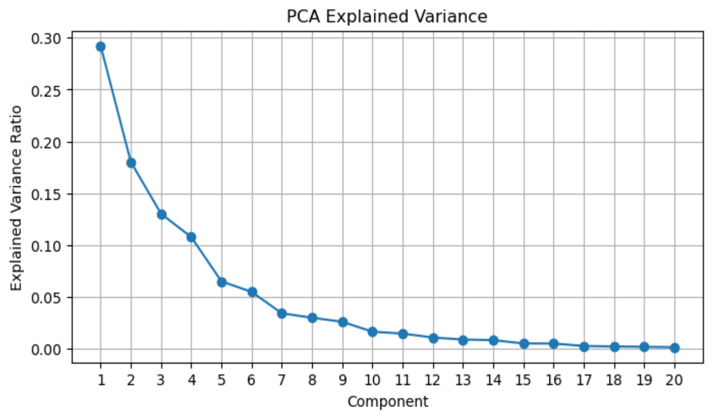
3. Data Preprocessing

- Removed target columns to prevent data leakage.
- Split the dataset into:
  - train dataset: used for model training.
  - test2 dataset: used for validation.
- Feature columns exclude non-numeric columns (date, name, obs_type).
- Scaling (Standardization using StandardScaler)
  - Each feature was standardized to have a **mean of 0** and **standard deviation of 1**.
  - This step ensures that PCA is not biased toward features with larger magnitudes.

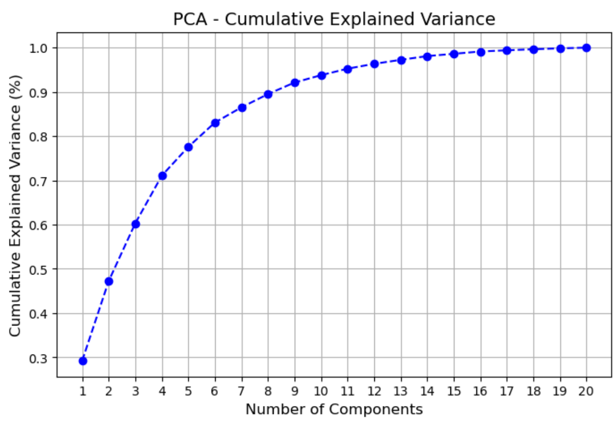
---

4. Principal Component Analysis (PCA)

- **Objective:** Dimensionality reduction and visualization of explained variance.
- **Model**: PCA was fitted on the **scaled training data** (X_train_scaled) and transformed both train and test datasets.
- **Components**: The analysis was performed using **20 principal components**.\
- **Plots:**
  - **Explained Variance Ratio Plot:**
    - ➢ Demonstrates how much variance each principal component captures.
    - ➢ This represents the proportion of the dataset's variance captured by each principal component.



    - ➢ Shows how much variance each principal component explains.
    - ➢ First few components capture most of the variance.
  - **Cumulative Explained Variance Plot:**



- ➢ Demonstrates how much total variance is explained as you add more components.
- ➢ The **first few components** capture the **majority of the variance**:
  - ✓ **1st component**: ~30%

- ✓ **First 5 components**: ~80% cumulative variance.
- ✓ **First 10 components**: ~95% cumulative variance.

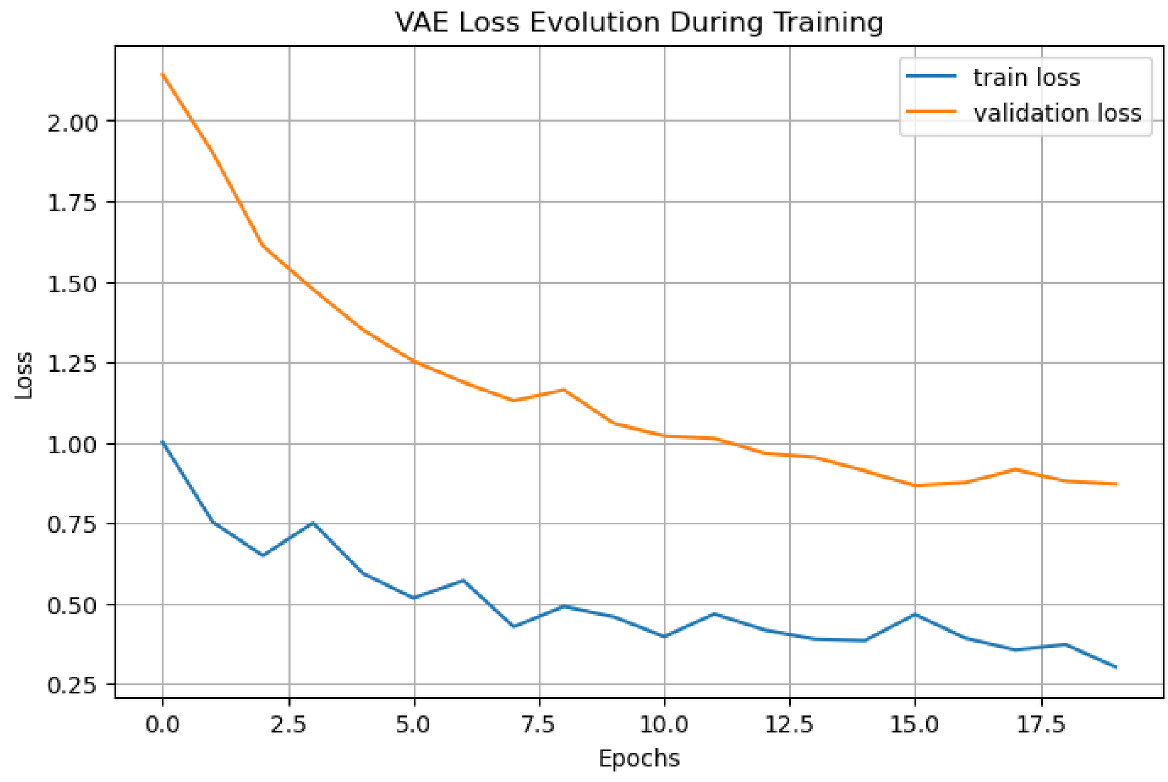
➢ After around **10 components**, the additional variance explained becomes marginal.

- ✓ Indicates **diminishing returns** in adding more components beyond the 10th.

---

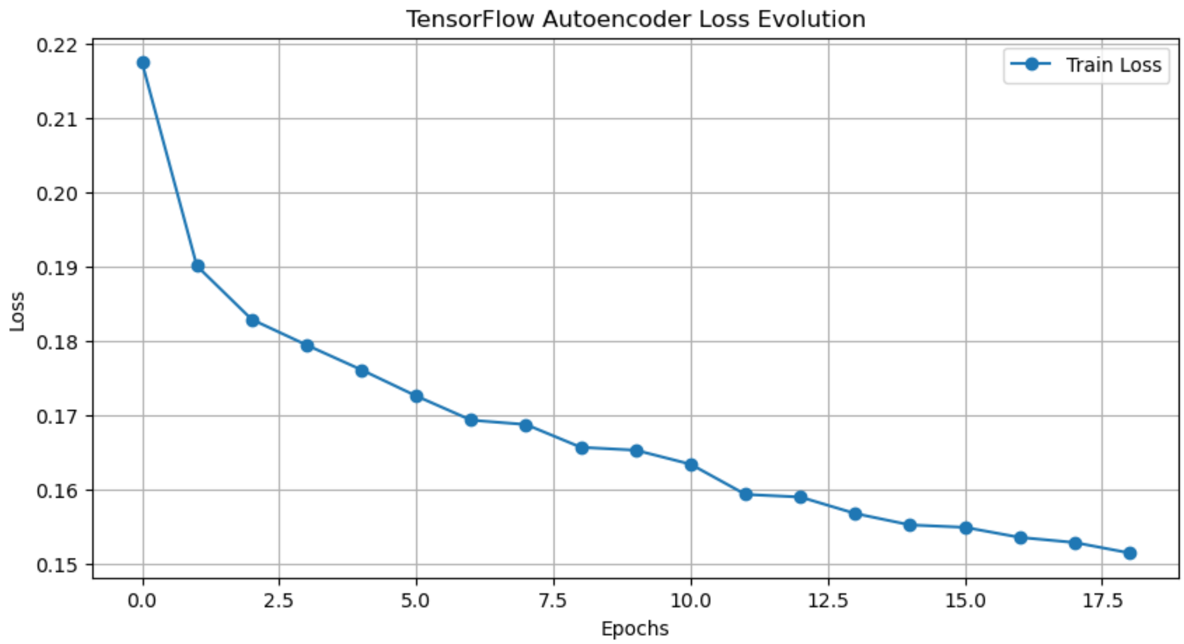
5. Variational Autoencoder (VAE)

- **Architecture**
    - **Encoder**
    - **Sampling Layer:** Lambda layer to sample from latent space.
    - **Decoder**
- **VAE Loss Function:**
    - **Reconstruction Loss** (MAE (Mean Absolute Error)
- **KL Divergence Loss:** Encourages latent space regularization.
- **Total Loss:** total_loss = reconstruction_loss + β * kl_loss
  (β = 0.0001)
- **Plots (VAE Loss Evolution During Training)**



6. TF Autoencoder

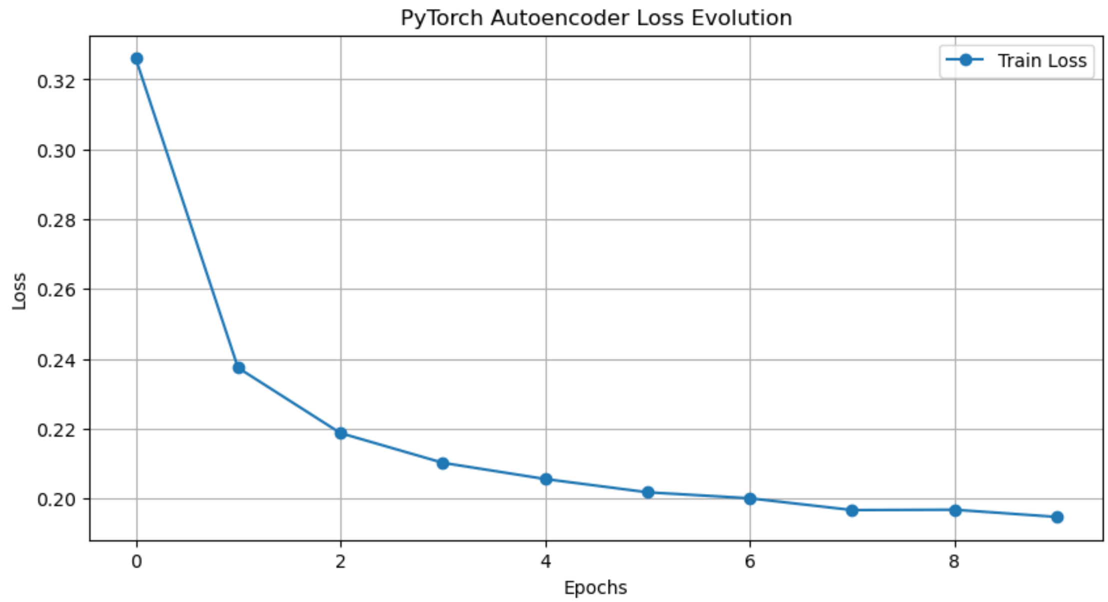
- **Architecture**
  - **Encoder**
  - **Decoder**
- **Loss Function**
  - **Reconstruction Loss** (MAE (Mean Absolute Error)
- **Plots (TF Autoencoder Loss Evolution During Training)**



---

7. PT Autoencoder

- **Architecture**
  - **Encoder**
  - **Decoder**
- **Loss Function**
  - **Reconstruction Loss** (MAE (Mean Absolute Error)
- **Plots (PT Autoencoder Loss Evolution During Training)**



---

8. reconstruction error on training vs. test data

model	Train Reconstruction Error	Test Reconstruction Error
VAE	0.0184	0.0848
TF Autoencoder	0.0659	0.2114
PT Autoencoder	0.0822	0.2415

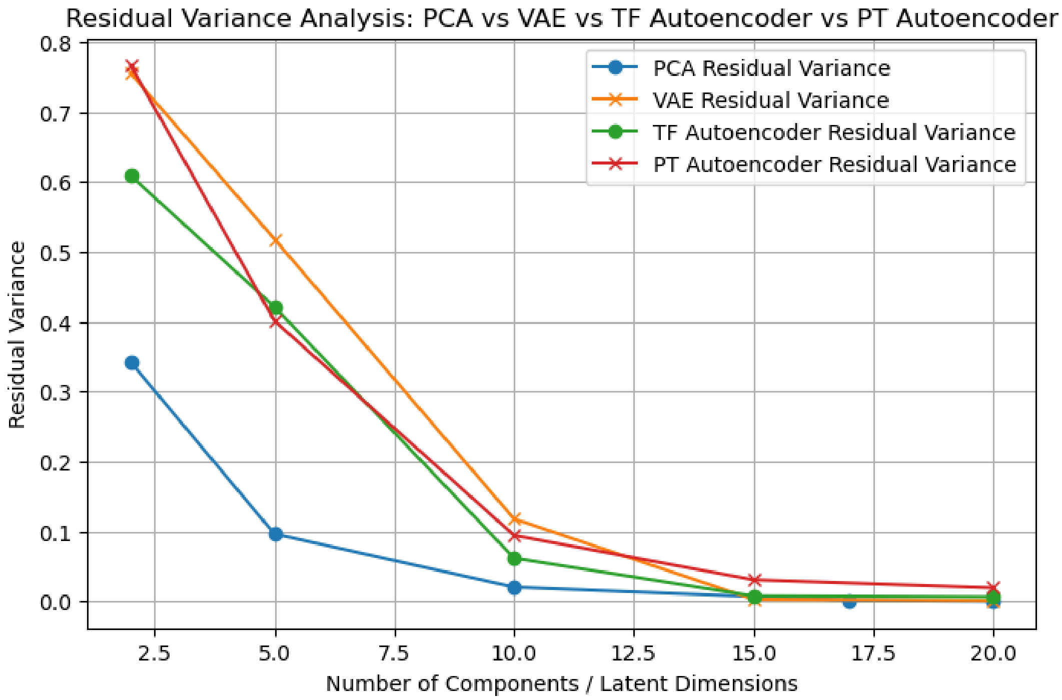
---

9. Residual Variance Analysis

Comparison of PCA vs VAE vs TensorFlow Autoencoder vs PyTorch Autoencoder

- **Purpose:**
  - evaluate and compare the **residual variance** (unexplained variance) of different dimensionality reduction techniques across varying numbers of components or latent dimensions.
- **Residual Variance Definition:**
  - **Residual Variance** = 1 - Explained Variance
  - Lower values are better: they indicate how much information from the original data remains **unexplained** by the reduced representation.

- **Methods Analyzed:**
  - **PCA (Principal Component Analysis)**
  - **VAE (Variational Autoencoder)**
  - **TensorFlow Autoencoder**
  - **PyTorch Autoencoder**
- **Plot**



---

10. Results and Discussion

- **PCA Analysis**
  - A total of 20 principal components were analyzed.
  - The cumulative explained variance reached nearly 100% with a reduced number of components, indicating that the majority of the dataset's variance is preserved with far fewer dimensions.
  - The first few components captured most of the variance, suggesting a strong potential for dimensionality reduction while retaining meaningful information

- **VAE**
  - Training loss decreased over time, confirming that the VAE successfully learned a compact representation.
  - Some fluctuations in loss suggest potential instability, which could be addressed through hyperparameter tuning or additional training epochs.
  - Compared to PCA, VAE provides a more flexible representation, potentially capturing non-linear relationships within the data.
- **TF Autoencoder**
  - The **TensorFlow model** achieved a **lower validation loss** compared to the PyTorch model.
  - **TensorFlow's learning rate** was **10x higher**, possibly accelerating convergence.
  - The **L2 regularizer** in TensorFlow might have helped **stabilize** the validation loss early.
  - **Batch size and architecture** were kept **consistent**, so the differences are mainly due to optimizer setup and library handling.
- **PT Autoencoder**
  - The model is learning well, showing a **clear reduction** in both train and validation loss.
  - **No signs of overfitting** within 10 epochs, though the slight increase in validation loss at later epochs suggests it might stabilize or benefit from early stopping/patience strategies.
  - Using **L1 loss** promotes robustness to outliers, making it suitable for your application if reconstruction smoothness is a priority.
- **Residual Variance Evaluation**

Method	**Residual Variance Trend**
PCA (Blue Circles)	Lowest residual variance overall. Sharp drop by 5 components, near zero by 10 components.
VAE (Orange X)	Starts with high variance (~0.75 at 2.5 dims), reduces gradually, still higher variance than PCA and TF Autoencoder across all points.
TF Autoencoder (Green Circles)	Starts better than VAE (~0.6 at 2.5 dims), converges close to zero by 15 components. Outperforms PT Autoencoder in most points.
PT Autoencoder (Red X)	Slightly worse than TF Autoencoder up to 10 components, but gets closer by 15-20 dimensions. Begins around ~0.75 residual variance.

8. Conclusion

- **PCA**
  - proved to be highly effective, with the first few principal components capturing most of the variance. The method provided an interpretable way to understand feature importance and dimensionality selection.
- **VAE**
  - The **training loss consistently decreased** over time, indicating that the VAE successfully **learned a compact and meaningful latent representation** of the data.
  - There were **some fluctuations** observed in the loss curve. This suggests potential **training instability**, which could be addressed by:
  - **Hyperparameter tuning** (adjusting learning rate, β for KL divergence, regularization).
  - **Increasing the number of training epochs** for better convergence.
  - **Batch size adjustments** or exploring alternative **optimizers**.
- **Autoencoder Training Loss Analysis**
  - **TensorFlow Autoencoder**
    - ✓ The **TensorFlow Autoencoder** achieved a **lower validation loss** compared to its PyTorch counterpart, suggesting **better reconstruction performance** under the given setup.
    - ✓ A **10x higher learning rate** in the TensorFlow model likely **accelerated convergence**, helping it to reach lower loss values **faster**. However, this also increases the risk of instability, which wasn't observed here due to other stabilizing factors.
    - ✓ The inclusion of an **L2 regularizer** in TensorFlow appears to have **stabilized validation loss early in training**, reducing the potential for overfitting and promoting generalization.
    - ✓ Since **batch size and architecture** were kept **identical** between TensorFlow and PyTorch implementations, the differences in performance are likely due to:
      - ❖ **Optimizer configuration** (learning rate, regularization)
      - ❖ **Framework-specific implementation nuances** in weight initialization and internal operations.

- **PyTorch Autoencoder**
  - ✓ The **PyTorch Autoencoder** shows **effective learning**, with both training and validation losses **steadily decreasing**, indicating **successful reconstruction capability**.
  - ✓ There were **no clear signs of overfitting** within the initial **10 epochs**, though a **slight increase in validation loss** toward the end suggests:
    - ❖ It may **stabilize with more training**.
    - ❖ Alternatively, applying **early stopping** or **patience strategies** could **prevent potential overfitting** in longer training runs.
  - ✓ The choice of **L1 loss (Mean Absolute Error)** in the PyTorch model encourages **robustness to outliers**, which can be particularly useful when **smooth reconstruction** is a **priority** in your application. L1 loss tends to produce **less blurry reconstructions**, though it may require more careful tuning for convergence.

▪ **Residual Variance Analysis**

- **10 components/latent dimensions** are generally **sufficient to retain most useful information**, as most models drop below **10% residual variance** by this point.
- If you need **very high fidelity** or you're working with sensitive tasks (like anomaly detection or medical data), **15 components** give you **even more complete information retention** (95%+ variance explained).