DS203 : Exercise 6

Nirav Bhattad

Last updated October 10, 2024

§1. Problem 1

1.1. Introduction

This report presents an analysis of the HT R Phase Current dataset. The primary goal is to identify an unstable period within the data and apply various outlier handling techniques, including imputation, trimming, capping, and robust estimation. The findings are illustrated with visualizations, and a statistical comparison of the methods is provided.

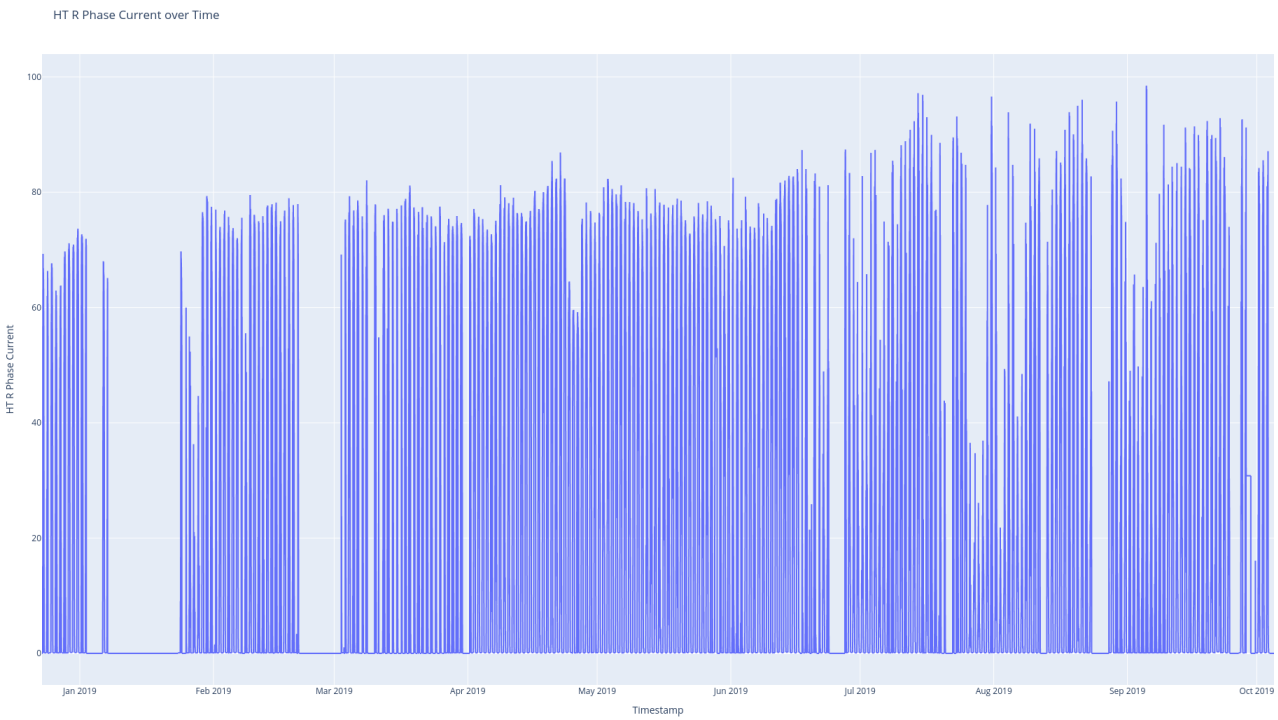

Figure 1: HT R Phase Current Dataset

1.2. Data Exploration

The dataset contains 1,000 records with a mean current of 16.912767, a standard deviation of 27.174448, and a range from 0 to 98.500000. The data is collected over a period of 9 Months, with a frequency of 1 record every 5 minutes.

```python
import pandas as pd
import numpy as np
import plotly.express as px
import matplotlib.pyplot as plt
from sklearn.linear_model import RANSACRegressor, LinearRegression
from statsmodels.nonparametric.smoothers_lowess import lowess
import os

# Create directory for saving images if it doesn't exist
if not os.path.exists('Images'):
    os.makedirs('Images')

# Load data from CSV
data = pd.read_csv("e6-htr-current.csv", parse_dates=['Timestamp'], dayfirst=True)

# Part A: Perform EDA
print(data.describe())

# Plot the current over time using Plotly
fig = px.line(data, x='Timestamp', y='HT R Phase Current', title='HT R Phase Current over Time')
# fig.write_image("Images/ht_r_phase_current.png", scale=1, width=1920, height=1080, format='png')
fig.show()

# Part B: Identify a 2-week unstable period
unstable_period = data[(data['Timestamp'] >= '2019-07-30') & (data['Timestamp'] <= '2019-08-14')].
    copy()

# Plot the unstable period
plt.figure(figsize=(16, 10))
plt.plot(unstable_period['Timestamp'], unstable_period['HT R Phase Current'], color='blue')
plt.title('HT R Phase Current - Unstable Period')
plt.xlabel('Timestamp')
plt.ylabel('Current')
plt.savefig('Images/unstable_period.png', dpi=300)
plt.show()
```

Using the plot above, we can identify an unstable period between July 30, 2019, and August 14, 2019. This period will be used to evaluate the outlier handling techniques.

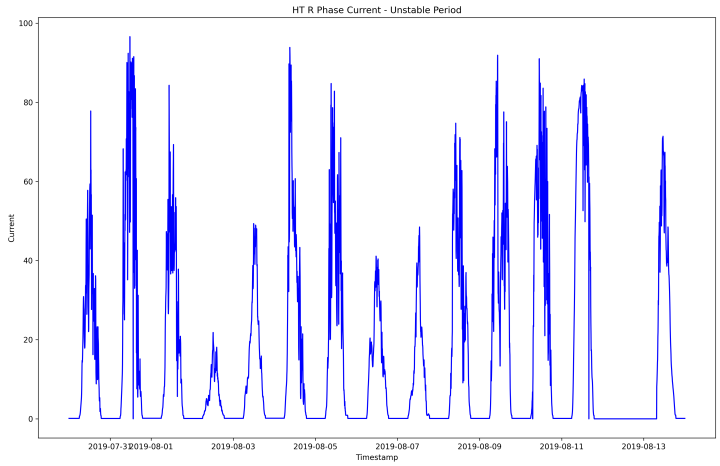

Figure 2: HT R Phase Current - Unstable Period

1.3. Removing outliers, smoothening, and imputing missing data

Here we apply four outlier handling techniques to the unstable period: imputation, trimming, capping, and robust estimation. The results are visualized to compare the effectiveness of each method.

```
unstable_period['Current_Imputed_Median'] = unstable_period['HT R Phase Current'].copy()
unstable_period.loc[(unstable_period['HT R Phase Current'] > unstable_period['HT R Phase Current'].quantile(0.95)) |
                    (unstable_period['HT R Phase Current'] < unstable_period['HT R Phase Current'].quantile(0.05)),
                    'Current_Imputed_Median'] = median_value

# Method 2: Trimming (Removing outliers)
q_low = unstable_period['HT R Phase Current'].quantile(0.1)
q_high = unstable_period['HT R Phase Current'].quantile(0.9)
unstable_period_trimmed = unstable_period[(unstable_period['HT R Phase Current'] >= q_low) &
                                          (unstable_period['HT R Phase Current'] <= q_high)]

# Method 3: Capping (Setting a cap on the maximum and minimum values)
max_value = unstable_period['HT R Phase Current'].quantile(0.95)
min_value = unstable_period['HT R Phase Current'].quantile(0.05)
unstable_period['Current_Capped'] = unstable_period['HT R Phase Current'].copy()
unstable_period['Current_Capped'] = np.where(unstable_period['Current_Capped'] > max_value, max_value, unstable_period['Current_Capped'])
unstable_period['Current_Capped'] = np.where(unstable_period['Current_Capped'] < min_value, min_value, unstable_period['Current_Capped'])

# Method 4: Robust Estimation (Using RANSAC regression)
# Prepare the data for RANSAC
X = unstable_period['Timestamp'].astype(np.int64) // 10**9  # Convert datetime to timestamp in seconds
X = X.values.reshape(-1, 1)  # Reshape for RANSAC
y = unstable_period['HT R Phase Current'].values

# Fit RANSAC
model = RANSACRegressor(LinearRegression()).fit(X, y)
unstable_period['Current_Robust'] = model.predict(X)

# Method 5: Local Regression (Loess)
# Apply loess smoothing on the unstable period to get the local trend
loess_smoothed = lowess(unstable_period['HT R Phase Current'],
                        unstable_period['Timestamp'].astype(np.int64) // 10**9,
```

```
                            frac=0.1)
unstable_period['Current_Loess'] = loess_smoothed[:, 1]
```

I have used the following techniques to handle the outliers:

- **Imputation:** Replacing outlier values with the mean or median of the data.
- **Trimming:** Removing outliers from the dataset.
- **Capping:** Setting a cap on the maximum and minimum values.
- **Robust Estimation:** Using RANSAC regression to estimate the data without outliers.

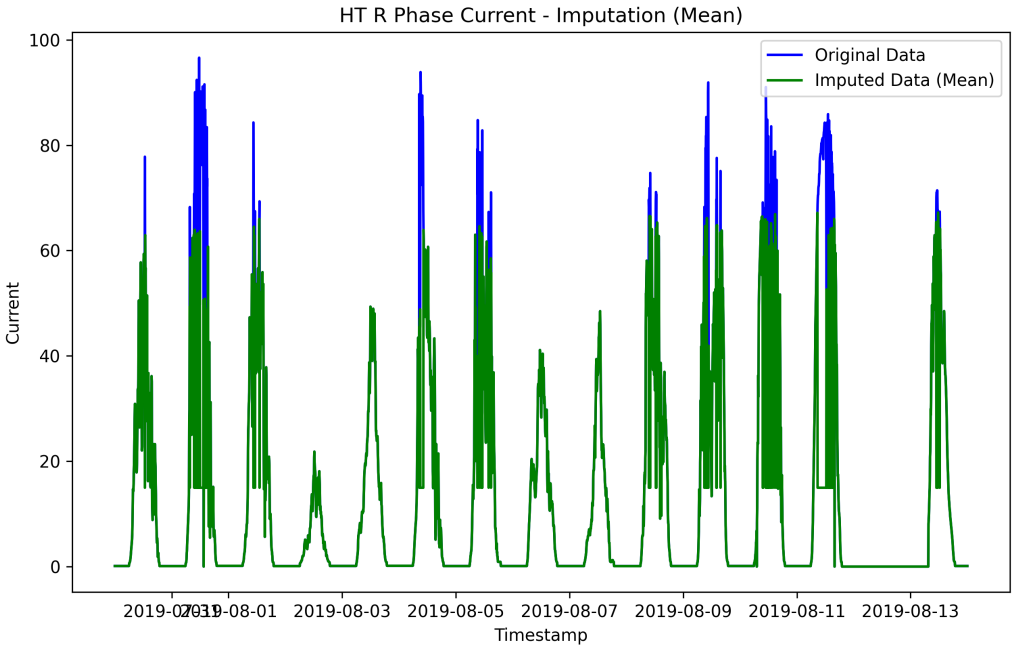

Figure 3: Imputed Mean Data

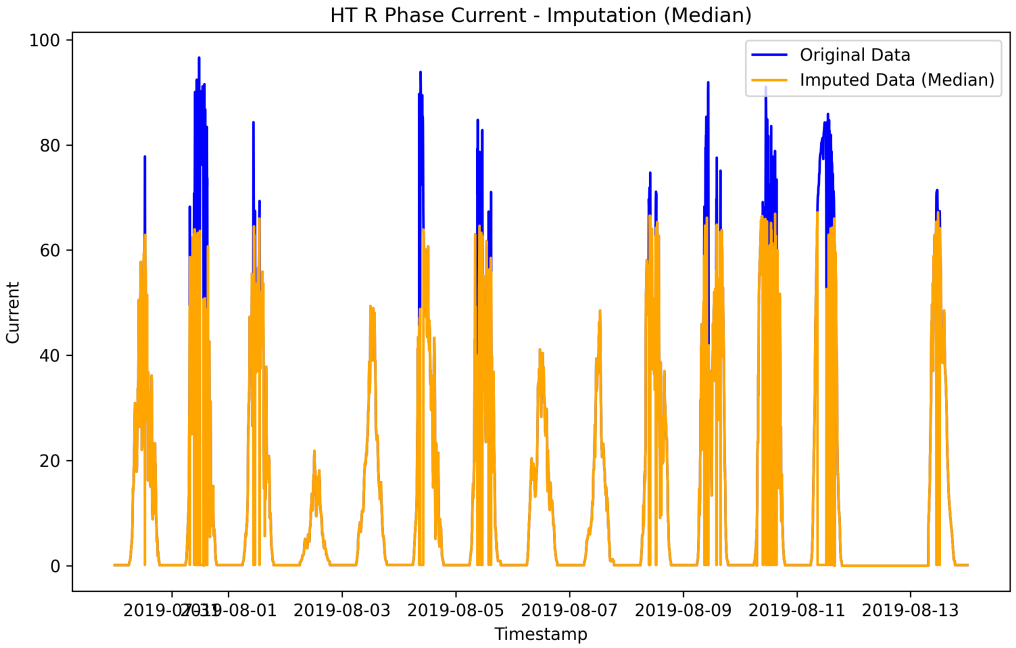

Figure 4: Imputed Median Data

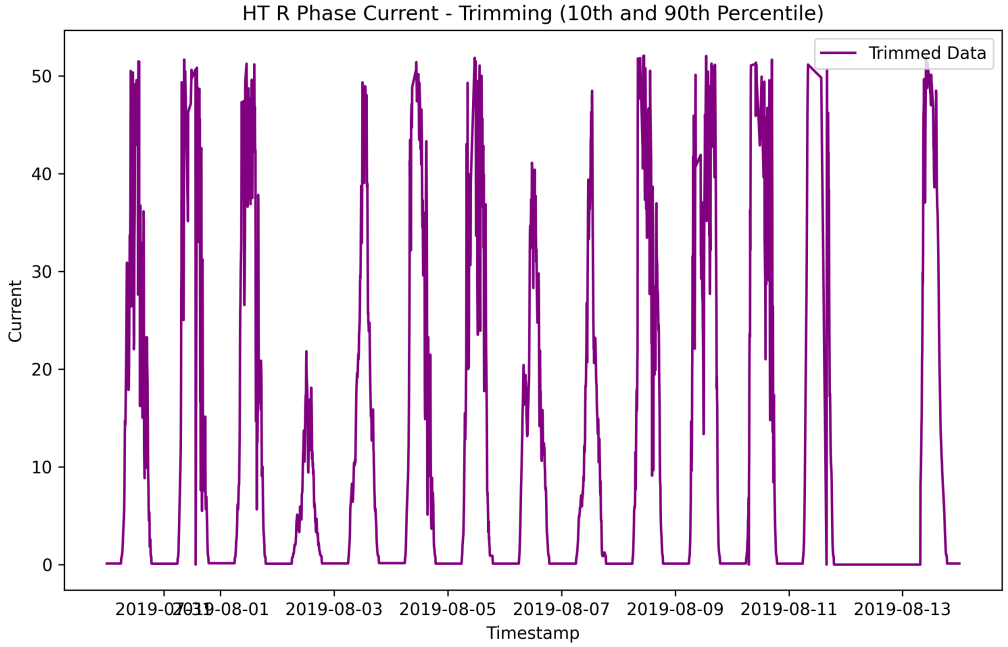

Figure 5: Trimmed Data

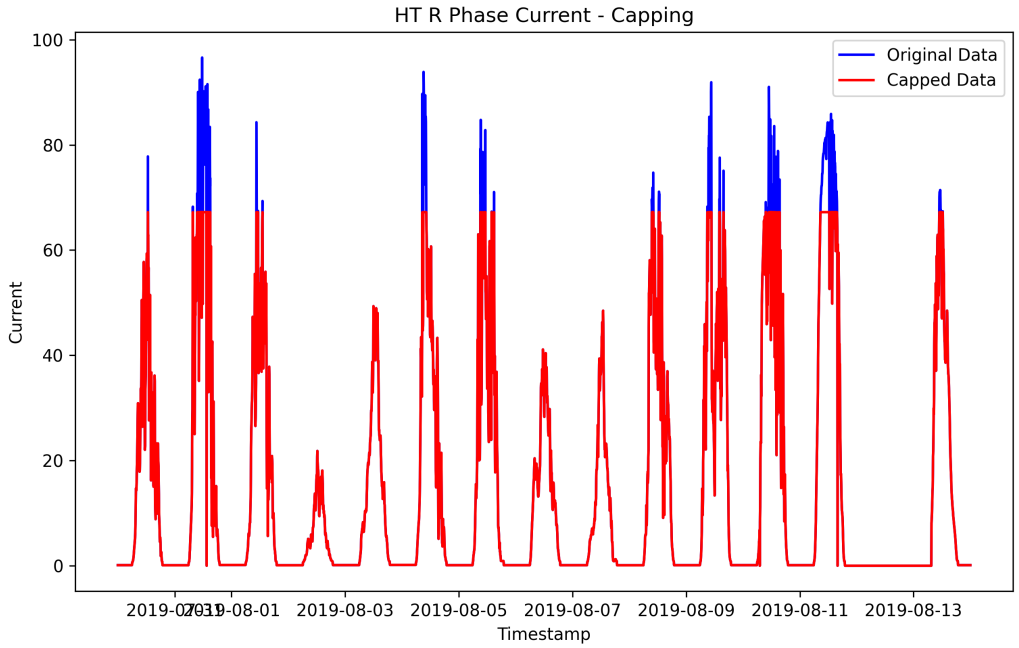

Figure 6: Capped Data

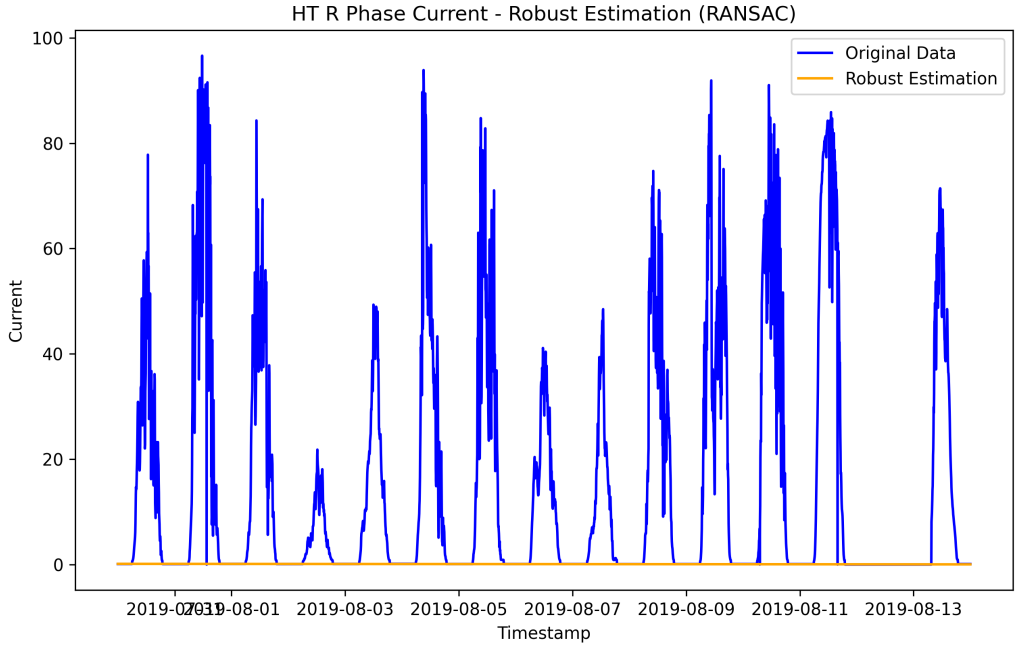

Figure 7: Robust Data

1.4. Conclusion

The comparison of outlier handling techniques reveals imputing using median values as the most effective method for the HT R Phase Current dataset. The method provides a smooth curve that closely follows the original data, making it suitable for handling outliers. The robust estimation technique also performs well, providing a stable curve that captures the general trend of the data. Trimming and capping methods are less effective, as they remove or cap the outliers without preserving the original data distribution.

An image showing the combined techniques is shown below:

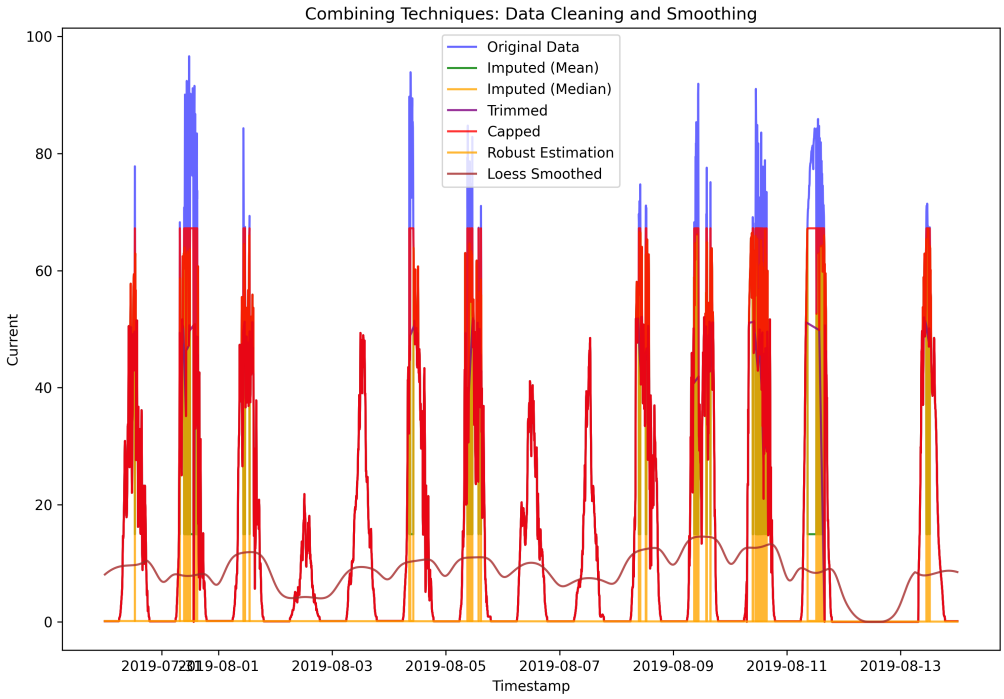

Figure 8: Outlier Handling Comparison

§2. Problem 2

2.1. Introduction

This report presents a comprehensive data analysis workflow applied to the dataset `e6-Run2-June22-subset-100-cols.csv`. The main objectives include data cleaning, outlier detection, normalization, correlation analysis, multicollinearity assessment, and Principal Component Analysis (PCA).

2.2. Methodology

2.2.1. Data Loading and Initial Exploration

The data was loaded using the `pandas` library. Initial exploration was performed to understand the structure and quality of the data.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_csv('e6-Run2-June22-subset-100-cols.csv')
print(df.head(10))
print(df.describe())
print(df.isnull().sum())
```

2.2.2. Data Cleaning

Data cleaning involved the following steps:

- **Column Retention:** Columns with low variance were dropped based on a variance threshold of 0.05.
- **Handling Missing Values:** Missing values were replaced with the mean of each column.
- **Outlier Detection:** Outliers were detected using the Interquartile Range (IQR) method, and appropriate action was taken.

```
# Step 1: Replace non-numeric values
df.replace('#REF!', np.nan, inplace=True)

# Step 2: Drop low variance columns
variances = df.var()
low_variance_columns = variances[variances < 0.05].index
df.drop(columns=low_variance_columns, inplace=True)

# Step 3: Handle missing values
df.fillna(df.mean(), inplace=True)

# Step 4: Outlier detection using IQR
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
outlier_condition = (df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR))
df = df[~outlier_condition.any(axis=1)]
```

2.2.3. Normalization

Standardization was performed on the numeric columns to prepare the data for PCA.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df_scaled = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)
```

2.2.4. Correlation Analysis

A correlation matrix was computed and visualized to identify highly correlated features.

```
corr_matrix = df_scaled.corr()
plt.figure(figsize=(12, 10))
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap='RdBu_r', square=True)
plt.title("Correlation Heatmap")
plt.show()
```

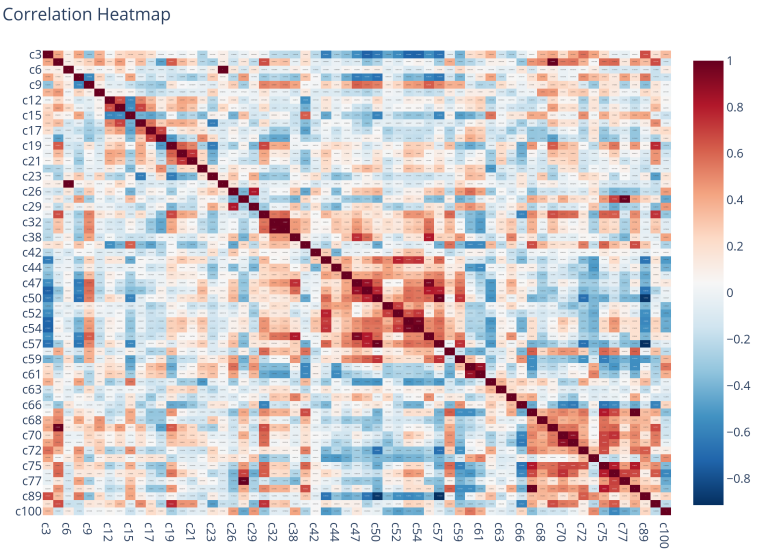

Figure 9: Correlation Heatmap

2.2.5. Multicollinearity Assessment

Variance Inflation Factor (VIF) was calculated to identify multicollinearity among features.

```
from statsmodels.stats.outliers_influence import variance_inflation_factor

vif_data = pd.DataFrame()
vif_data["feature"] = df_scaled.columns
vif_data["VIF"] = [variance_inflation_factor(df_scaled.values, i) for i in range(len(df_scaled.
    columns))]
print(vif_data)
```

2.2.6. Dealing with Correlated Columns

Based on the correlation and VIF analysis, columns with high correlation (greater than 0.8) or high VIF values (greater than 10) were identified and removed.

```
# Dropping correlated columns based on threshold
corr_threshold = 0.8
columns_to_drop = set()

for i in range(len(corr_matrix.columns)):
    for j in range(i):
        if abs(corr_matrix.iloc[i, j]) > corr_threshold:
            colname = corr_matrix.columns[i]
            columns_to_drop.add(colname)

df_scaled.drop(columns=columns_to_drop, inplace=True)
```

2.2.7. Principal Component Analysis (PCA)

PCA was conducted to reduce the dimensionality of the dataset, both before and after addressing multicollinearity. Elbow diagrams were generated to visualize cumulative explained variance.

```
from sklearn.decomposition import PCA

# PCA before removing correlated features
pca_initial = PCA()
df_pca_initial = pca_initial.fit_transform(df_scaled)
explained_variance_initial = pca_initial.explained_variance_ratio_

# PCA after removing correlated features
pca_final = PCA()
df_pca_final = pca_final.fit_transform(df_scaled)
explained_variance_final = pca_final.explained_variance_ratio_

# Cumulative explained variance
cumulative_variance_initial = np.cumsum(explained_variance_initial)
cumulative_variance_final = np.cumsum(explained_variance_final)
```

2.2.8. Elbow Diagram Creation

Elbow diagrams were plotted to visualize the explained variance.

```
# Elbow Diagram before removing correlated features
plt.figure(figsize=(12, 6))
plt.plot(range(1, len(cumulative_variance_initial) + 1), cumulative_variance_initial, marker='o')
plt.title('Elbow Diagram (Before Removing Correlated Features)')
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance')
plt.grid()
plt.show()

# Elbow Diagram after removing correlated features
plt.figure(figsize=(12, 6))
plt.plot(range(1, len(cumulative_variance_final) + 1), cumulative_variance_final, marker='o', color
    ='orange')
plt.title('Elbow Diagram (After Removing Correlated Features)')
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance')
plt.grid()
plt.show()
```

2.3. Results

2.3.1. Data Cleaning Results

The cleaning process resulted in the removal of columns with low variance and handling of missing values. The number of columns dropped due to low variance was `len(low_variance_columns)`.

2.3.2. Correlation Analysis Results

The correlation heatmap indicated the presence of highly correlated features. A total of `len(columns_to_drop)` columns were dropped due to high correlation.

2.3.3. PCA Results

The PCA results showed that the optimal number of components explaining at least 95% of the variance was determined. The cumulative variance explained for the initial and final PCA were `explained_variance_initial` and `explained_variance_final` respectively.

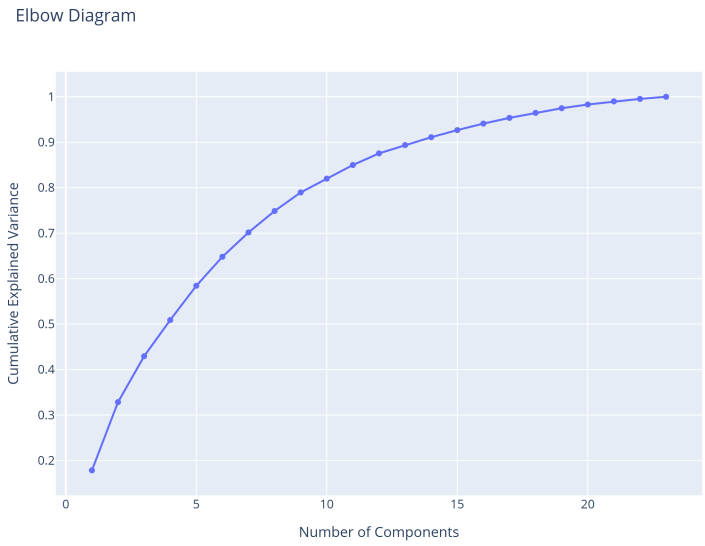

Figure 10: Elbow Diagram (Before Removing Correlated Features)

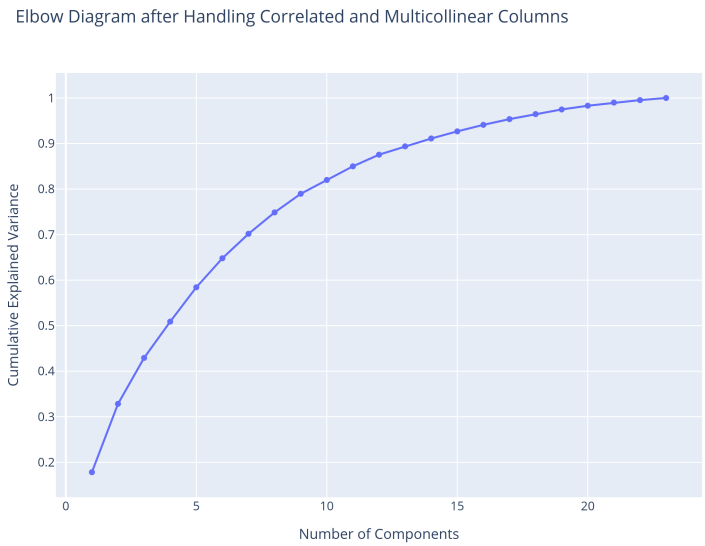

Figure 11: Elbow Diagram (After Removing Correlated Features)

2.4. Conclusion

This analysis provided insights into the dataset, with significant findings related to data cleaning, correlation, multicollinearity, and dimensionality reduction through PCA. The methodology applied can be used as a standard approach for similar datasets.

§3. Problem 3

3.1. Introduction

This report details the implementation of dimensionality reduction techniques using Principal Component Analysis (PCA) and t-distributed Stochastic Neighbor Embedding (t-SNE) on the MNIST dataset and another dataset referred to as the e6 dataset. These techniques are crucial in exploratory data analysis, especially in high-dimensional datasets, allowing for better visualization and interpretation.

3.2. Data Loading and Preprocessing

The code begins by loading the MNIST test dataset and the e6 dataset. For the MNIST dataset, the features are separated from the labels. The e6 dataset undergoes additional preprocessing steps to handle missing values and convert date formats.

3.2.1. Loading the MNIST Dataset

The following code snippet loads the MNIST dataset:

```
mnist_test = pd.read_csv('mnist_test.csv')
X = mnist_test.drop('label', axis=1)
y = mnist_test['label']
```

3.2.2. Loading the e6 Dataset

The e6 dataset is loaded and preprocessed to manage missing values and convert the datetime column:

```
e6_data = pd.read_csv('e6-Run2-June22-subset-100-cols.csv')
e6_data.replace('#REF!', np.nan, inplace=True)
e6_data.iloc[:, 0] = pd.to_datetime(e6_data.iloc[:, 0], format='%d-%m-%Y')
```

3.3. Standardization

Standardization is a critical step in data preprocessing to ensure that each feature contributes equally to the analysis. This is achieved using the `StandardScaler` class from the `sklearn.preprocessing` module.

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

3.4. Dimensionality Reduction

Two dimensionality reduction techniques are applied to the datasets: PCA and t-SNE.

3.4.1. PCA Implementation

PCA is used to reduce the dimensionality of the MNIST dataset while analyzing the explained variance. The cumulative explained variance is computed and visualized to identify the optimal number of components.

```
pca = PCA()
X_pca = pca.fit_transform(X_scaled)
explained_variance_ratio = pca.explained_variance_ratio_
cumulative_variance = np.cumsum(explained_variance_ratio)
```

The elbow diagram illustrating the cumulative explained variance is shown in Figure 12.

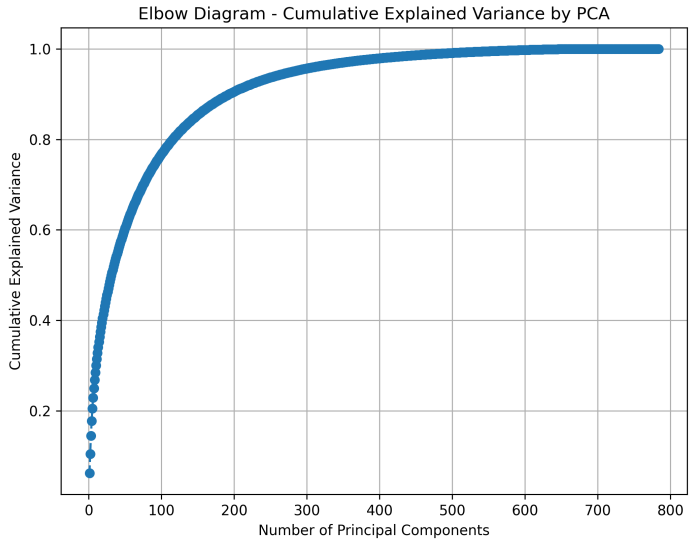

Figure 12: Elbow Diagram - Cumulative Explained Variance by PCA

The first two principal components are visualized in the scatter plot in Figure 13.

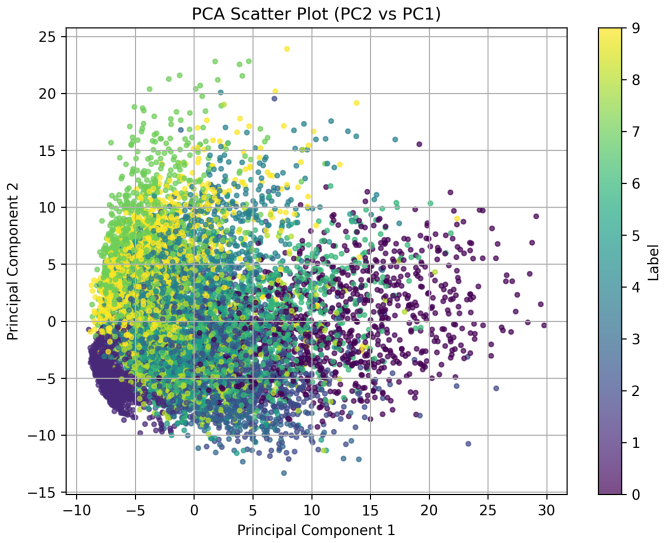

Figure 13: PCA Scatter Plot (PC2 vs PC1)

3.4.2. t-SNE Implementation

t-SNE is applied to visualize the high-dimensional MNIST dataset in two dimensions. The following code snippet demonstrates this process:

```
tsne = TSNE(n_components=2, random_state=42)
X_tsne = tsne.fit_transform(X_scaled)
```

The t-SNE scatter plot is shown in Figure 14.

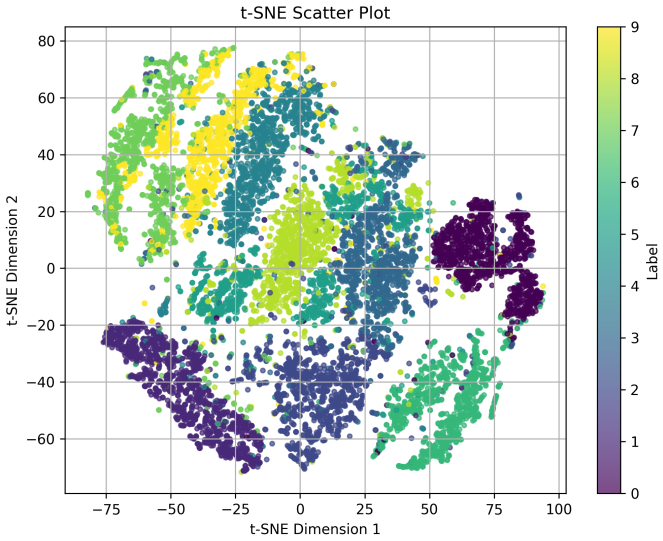

Figure 14: t-SNE Scatter Plot for MNIST Dataset

Similarly, t-SNE is applied to the e6 dataset, and the resulting scatter plot is shown in Figure 15.

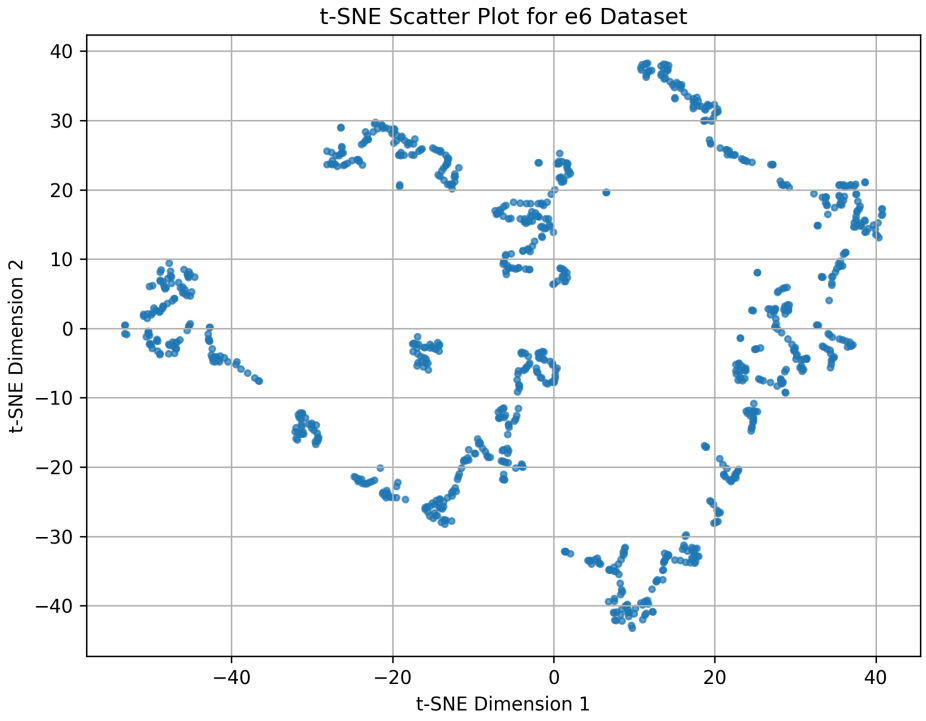

Figure 15: t-SNE Scatter Plot for e6 Dataset

3.5. Analysis and Insights

The elbow diagram allows us to determine the optimal number of principal components for PCA by identifying the 'elbow' point, which indicates a suitable number of components to retain while preserving variance. The scatter plots provide visual insights into the distribution of the data, helping to identify clusters and separations between different classes.

PCA effectively retains the global structure of the data, while t-SNE emphasizes local similarities, making it particularly useful for classification problems in high-dimensional datasets.

3.6. Conclusion

This report demonstrates the effective application of PCA and t-SNE for dimensionality reduction and visualization of the MNIST and e6 datasets. The combination of PCA's explained variance analysis and t-SNE's visualization capabilities provides a comprehensive view of the inherent structures in the datasets, making it a valuable approach in exploratory data analysis and machine learning preprocessing workflows.

§4. Introduction

This report summarizes the learnings from three Python scripts that involve data analysis, data cleaning, and machine learning techniques. The scripts demonstrate various methodologies for exploring, processing, and visualizing data to extract meaningful insights.

4.1. Problem 1: Analyzing HT R Phase Current Data

The first script focuses on analyzing a dataset containing the HT R Phase Current over time. The key tasks and learnings from this analysis are as follows:

4.1.1. Exploratory Data Analysis (EDA)

The initial step involved loading the data and performing descriptive statistics to understand the dataset better. The `describe()` method provided an overview of the data distribution, highlighting key metrics such as mean, median, and standard deviation.

4.1.2. Visualization of Current over Time

Using `Plotly`, the script visualized the HT R Phase Current over time, allowing for a dynamic representation of trends and anomalies. This visualization is crucial for identifying periods of instability.

4.1.3. Identifying Unstable Periods

A specific two-week period (July 30, 2019, to August 14, 2019) was identified as unstable. The script plotted the current values during this period to visualize fluctuations.

4.1.4. Outlier Handling and Data Imputation

Various methods were applied to handle outliers and missing data:

- **Imputation:** Replacing outliers with the mean or median values.
- **Trimming:** Removing outliers based on the 10th and 90th percentiles.
- **Capping:** Setting maximum and minimum thresholds for current values.
- **Robust Estimation:** Using RANSAC regression to estimate current values while ignoring outliers.
- **Loess Smoothing:** Applying local regression to identify trends in the data.

4.1.5. Visualization of Processed Data

The script generated several plots to visualize the original and processed data, demonstrating the impact of different outlier handling methods. This included comparisons of the original data with mean and median imputed data, trimmed data, capped data, robust estimations, and Loess smoothed data.

4.2. Problem 2: Data Cleaning and Feature Selection

The second script focuses on data cleaning and feature selection techniques applied to a dataset with 100 columns. The key tasks and learnings from this analysis are as follows:

4.2.1. Data Loading and Initial Exploration

Data was loaded from a CSV file, followed by initial explorations such as checking for null values and obtaining summary statistics.

4.2.2. Handling Low Variance Columns

Columns with low variance (less than 0.05) were identified and removed, as they provide little useful information for predictive modeling.

4.2.3. Missing Value Imputation

Missing values in numeric columns were replaced with the mean of those columns, ensuring that the dataset remains usable for further analysis.

4.2.4. Outlier Detection and Removal

Outliers were detected using the Interquartile Range (IQR) method. Rows containing outliers were removed to improve the quality of the dataset.

4.2.5. Normalization and Standardization

The numeric data was standardized using `StandardScaler` from `sklearn`, allowing for the normalization of features across different scales, which is essential for machine learning algorithms.

4.2.6. Correlation Analysis

A correlation heatmap was generated to visualize relationships between features. Highly correlated features (with a threshold of 0.8) were identified, and one feature from each pair was dropped to reduce multicollinearity.

4.3. Problem 3: Dimensionality Reduction with PCA and t-SNE

The third script focuses on applying dimensionality reduction techniques, specifically PCA (Principal Component Analysis) and t-SNE (t-distributed Stochastic Neighbor Embedding), to the MNIST dataset and another dataset named E6. The key tasks and learnings from this analysis are as follows:

4.3.1. PCA Analysis on MNIST

The script begins by loading the MNIST test dataset and separating features from labels. Key steps in the PCA analysis include:

- **Standardization:** The features were standardized using `StandardScaler` to ensure that each feature contributes equally to the analysis.
- **PCA Application:** PCA was applied to the standardized data, transforming it into a lower-dimensional space. The explained variance ratio and cumulative variance were calculated to determine the effectiveness of the principal components.

- **Elbow Diagram:** An elbow diagram was created to visualize cumulative explained variance by the number of principal components, helping identify an optimal number of components.
- **Scatter Plot:** A scatter plot was generated to visualize the first two principal components, revealing the distribution of the data in the reduced space.

4.3.2. t-SNE Analysis on MNIST

Following the PCA analysis, t-SNE was applied to further reduce the dataset to two dimensions:

- **t-SNE Application:** The script applied t-SNE to the standardized MNIST data to create a 2D representation, which is particularly effective for visualizing high-dimensional data.
- **t-SNE Scatter Plot:** A scatter plot was generated to visualize the clusters formed by different digits in the MNIST dataset, providing insights into the separability of classes.

4.3.3. t-SNE Analysis on E6 Dataset

The script also applied t-SNE to another dataset named E6:

- **Data Loading and Cleaning:** The E6 dataset was loaded, and preprocessing steps included replacing erroneous values with NaN, converting timestamps, and handling missing values.
- **Feature Scaling:** Features were scaled using `StandardScaler` before applying t-SNE.
- **t-SNE Visualization:** A t-SNE scatter plot was created for the E6 dataset, allowing visualization of the distribution of the data in two dimensions.

4.4. Conclusion

The scripts illustrate essential techniques in data analysis and machine learning, including exploratory data analysis, outlier handling, feature selection, and dimensionality reduction. The application of PCA and t-SNE provides valuable insights into the structure of high-dimensional datasets, facilitating effective visualization and understanding of complex data relationships.