HOME PROJECT REPORT

On the course

DATA ANALYSIS

by

Sharmi Islam

Supervisor

D.Sc., Prof.

Mirkin Boris Grigorievich

HIGHER SCHOOL OF ECONOMICS

Faculty of Computer Science

Moscow 2023

Abstract

The project aimed to explore various data analysis methodologies to gain comprehensive insights into the selected dataset, focusing not only on predicting values but also understanding observed patterns. Several analytical methods were employed, including correlation coefficient, bootstrapping, principal component analysis (PCA), singular value decomposition (SVD), and linear regression.

The investigation began with a thorough exploration of the dataset's features, emphasizing the correlation coefficient to identify relationships between different attributes. Bootstrapping techniques were utilized to derive confidence intervals, particularly to estimate the grand mean and assess cluster feature means relative to this grand mean.

Cluster analysis served as a pivotal tool to identify distinct patterns among entities within the dataset, allowing the segmentation of data into coherent groups based on similarities. Interpretations were drawn by analyzing differences between cluster feature means and the overall grand mean, shedding light on the distinct characteristics of each cluster.

Contingency tables were employed to examine associations between various features, enabling the computation of average Quetelet coefficients to test for feature independence. Principal Component Analysis (PCA) was leveraged to reduce data dimensions while retaining essential information, revealing significant patterns with different standardization methods. Additionally, Singular Value Decomposition (SVD) was compared to conventional PCA outcomes to assess consistency.

A pair of features exhibited a linear relationship, enabling predictive modeling and estimation of mean absolute relative error values. This analysis not only focused on prediction accuracy but also emphasized understanding underlying data patterns and

relationships, providing a holistic view of the dataset's characteristics.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The climate change dataset encapsulates a compilation of observations and measurements regarding global temperature trends over an extensive period. It encompasses a spectrum of variables, including land and ocean temperature averages, temperature uncertainties, maximum and minimum temperature records, and more. This dataset serves as a fundamental resource for studying climate variations, trends, and potential implications on our environment. Analyzing this dataset provides insights into temperature fluctuations, potential patterns, and aids in understanding the broader context of climate change phenomena.

1.1 Programs Used

To solve all the tasks, programs have been used and written in the programming interpretable language Python 3. All the code is supplied with the comments. All the code scripts are on the APPENDIX.

1.2 Dataset Overview

To simplify this project here I used Global Temperatures dataset from kaggle which include the features of ,

- dt
- Land Average Temperature

- Land Average Temperature Uncertainty
- Land Max Temperature
- Land Max Temperature Uncertainty
- Land Min Temperature
- Land Min Temperature Uncertainty
- Land And Ocean Average Temperature
- Land And Ocean Average Temperature Uncertainty

  This features figure 1 give us a clear view on the average temperature throw out the time period.

```
[2]
                dt  LandAverageTemperature  LandAverageTemperatureUncertainty  \
1200  1850-01-01                   0.749                              1.105
1201  1850-02-01                   3.071                              1.275
1202  1850-03-01                   4.954                              0.955
1203  1850-04-01                   7.217                              0.665
1204  1850-05-01                  10.004                              0.617
...          ...                     ...                                ...
3187  2015-08-01                  14.755                              0.072
3188  2015-09-01                  12.999                              0.079
3189  2015-10-01                  10.801                              0.102
3190  2015-11-01                   7.433                              0.119
3191  2015-12-01                   5.518                              0.100

      LandMaxTemperature  LandMaxTemperatureUncertainty  LandMinTemperature  \
1200               8.242                          1.738              -3.206
1201               9.970                          3.007              -2.291
1202              10.347                          2.401              -1.905
1203              12.934                          1.004               1.018
1204              15.655                          2.406               3.811
...                  ...                            ...                 ...
3187              20.699                          0.110               9.005
3188              18.845                          0.088               7.199
3189              16.450                          0.059               5.232
3190              12.892                          0.093               2.157
3191              10.725                          0.154               0.287

      LandMinTemperatureUncertainty  LandAndOceanAverageTemperature  \
1200                          2.822                          12.833
1201                          1.623                          13.588
1202                          1.410                          14.043
1203                          1.329                          14.667
1204                          1.347                          15.507
...                             ...                             ...
3187                          0.170                          17.589
3188                          0.229                          17.049
3189                          0.115                          16.290
3190                          0.106                          15.252
3191                          0.099                          14.774

      LandAndOceanAverageTemperatureUncertainty
1200                                      0.367
1201                                      0.414
1202                                      0.341
1203                                      0.267
1204                                      0.249
...                                         ...
3187                                      0.057
3188                                      0.058
3189                                      0.062
3190                                      0.063
3191                                      0.062

[1992 rows x 9 columns]
```

Figure 1.1: Dataset View

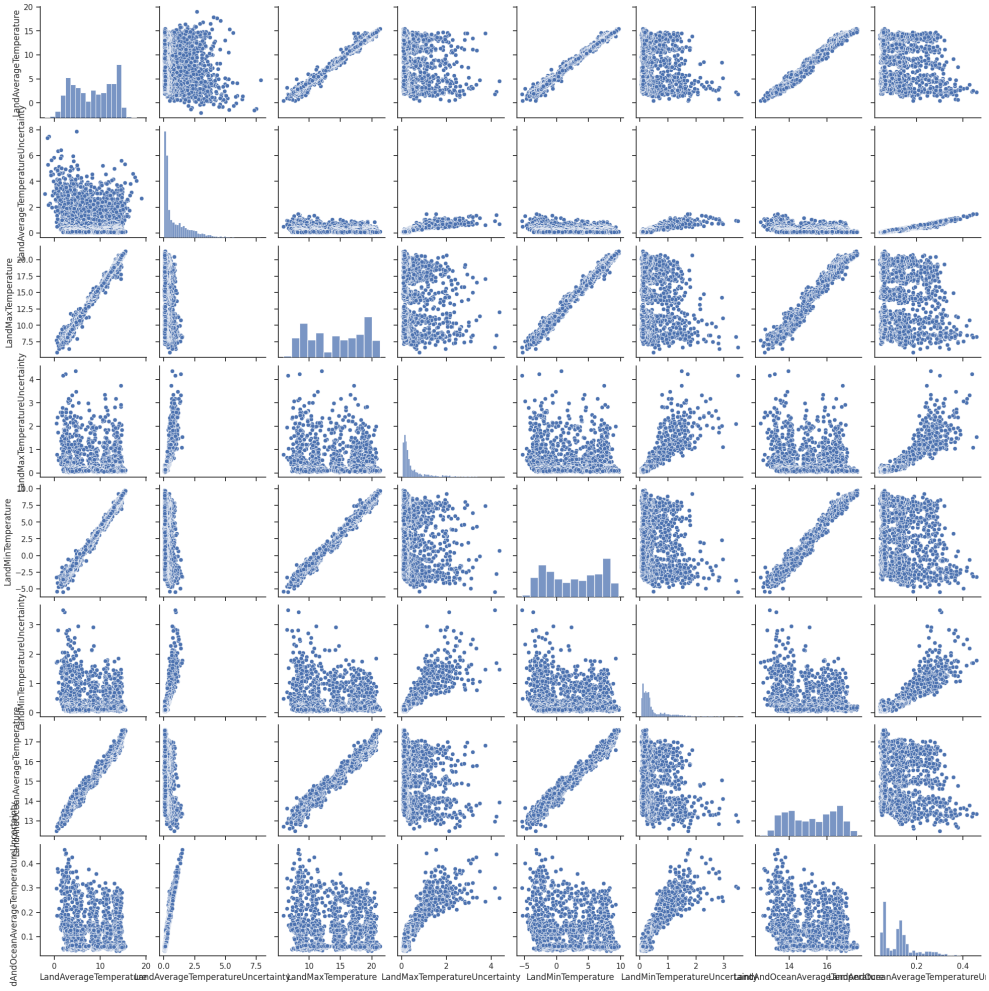


Figure 1.2: The pairwise plots

Chapter 2

Correlation Coefficient

The analysis of two features on one dataset may be explained in case of the correlation between them. That characteristic may be useful in the prediction problems, studying the information behind these features. As it can be seen on the Figure 1.2, there are features in the dataset that represent a linear dependency.For this project purpouse I have selected ”LandAverageTemperature” as a independent variable and ”LandMaxTemperature” as a dependent variable.

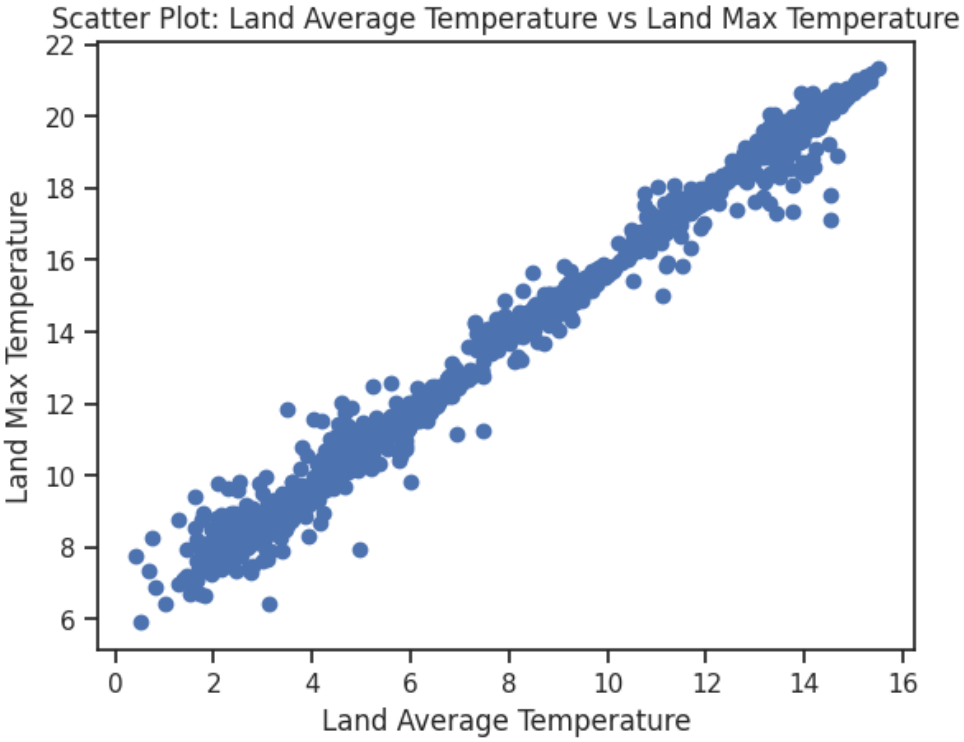


Figure 2.1: scatter plot

2.1 Linear Regression

Theoritically if X increase and y also increase we can state its a positive linear correlation.

To build the linear regression model,Here LinearRegression from the package sklearn was used. The function used has method fit, which is called with values of the selected features.

The results obtained: LandMaxTemperature = 0.59 * LandAverageTemperature + 9.35. For every increasing unit of Land Average Temperature the Max temperature will increase by 0.59 unit.

. The full code for that chapter is presented in APPENDIX .

2.2 correlation and determinacy coefficients

The slope of regression model is positive and shows a direct relation between two features. The correlation coefficient of 0.59 indicates a moderate positive linear relationship between the 'LandAverageTemperature' and 'LandMaxTemperature' features. It suggests that as one variable increases, the other tends to increase as well, but the relationship isn't perfectly linear.

The determinacy coefficient (R-squared) of 0.58 figure 2.2 implies that around 58 percent of the variability in 'LandMaxTemperature' can be explained by 'LandAverageTemperature'. This indicates a moderate level of predictability of the 'LandMaxTemperature' based on 'LandAverageTemperature'.

In summary, while there's a noticeable relationship between the two features, the predictability of 'LandMaxTemperature' solely based on 'LandAverageTemperature' isn't exceptionally high, indicating the presence of other factors influencing 'LandMaxTemperature'.

For predicting random values of LandMaxTemperature 4 random values of LandAverageTemperature were chosen. The predicted and real values are demonstrated in Table 2.1.

As it can be noticed, the values differ slightly. To measure the error, mean relative absolute error (MRAE) was calculated in two different approaches. The values are

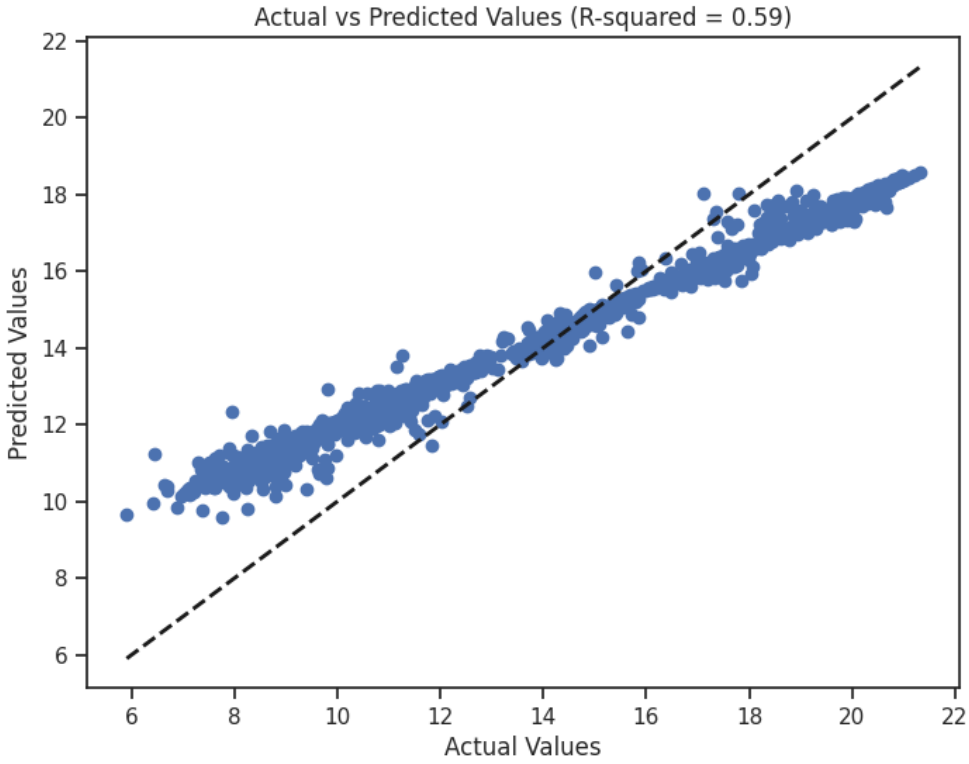


Figure 2.2: scatter plot

Predicted value	True value	Differences
9.79	8.24	1.55
11.18	9.97	1.21
12.30	10.34	2.04
13.65	12.93	0.72

Table 2.1: Predicted and true values

presented in Table 2.2. They come identical Mean Relative Absolute Error (MRAE) values for both the Deterministic Algorithmic (DA) and Machine Learning (ML) approaches imply an intriguing outcome. This scenario is quite unusual because it suggests that, in this specific context and with the provided dataset, both methods yield nearly equivalent predictive accuracy. Such parity in performance between these distinct approaches could indicate that either both methods are equally effective for this particular task or that the nature of the dataset is such that it doesn't significantly favor one method over the other. Further analysis or exploration might reveal more insights into why these methods display such similar predictive performance in this context.

The provided correlation coefficient suggests a moderately strong positive linear relationship between the predictors and the target variable. However, the determinacy

Error	Value
Mean relative absolute error(DA)	0.14
Mean relative absolute error(ML)	9.97

Table 2.2: Errors of linear regression

coefficient (R^2) indicates that approximately 58.8 percent of the variance in the target variable can be explained by the predictors in the model.

Interestingly, the Mean Relative Absolute Error (MRAE) for both the Deterministic Algorithmic (DA) and Machine Learning (ML) approaches is identical, suggesting an equivalent level of predictive accuracy between these methods. It's noteworthy that despite the similarity in MRAE values, the determinacy coefficient (R^2) doesn't necessarily align with the predictive performance as assessed by MRAE. This discrepancy might indicate that while the predictors explain a moderate portion of the variance, the accuracy of predicting specific values might not reflect that model's overall explanatory power. Further investigation into the data and model might help reconcile these results and determine other factors influencing the model's performance.

Chapter 3

PCA/SVD

In this part the selected dataset was selected in subset of 3-6 features related to the same aspec. potential subset of features related to temperature in a climate dataset:

1. LandAverageTemperature: Represents the average temperature over land areas.
2. LandMaxTemperature: Denotes the maximum temperature over land areas.
3. LandMinTemperature: Represents the minimum temperature over land areas.
4. LandAndOceanAverageTemperature: Indicates the average temperature over both land and ocean areas.

These features belong to the same aspect of temperature measurements across different regions and domains (land, ocean, combined), making them a coherent subset for analysis. They offer a range of temperature-related insights, from overall averages to specific maximum and minimum values, and cover both land and ocean areas for a more comprehensive view of temperature variations.

For Range Normalization percentage Contribution of Principal Components: PC1: 99.29 percent , PC2: 0.47 percent, PC3: 0.16 percent, PC4: 0.08 percent

Normalization Technique	Observations	Interpretation
Range Normalization	Clusters slightly overlapped, scattered points	Less distinct grouping
Z-Score Standardization	Compact clusters, distinct groups	Clearer separation

Table 3.1: Comparison of Visualization Observations

Data plotted using PC1 and PC2 from PCA after range normalization figure 3.1. Th clusters are discernible but might exhibit overlapping in certain areas.Data points appear scattered across the plot but show a semblance of grouping.For Z-Score Standardization The clusters seem more compact and distinct compared to the range normalization table 3.2. Data points are comparatively more separated and form clearer groupings.Z-score standardization appears to offer better cluster separation and clearer groupings in the visualization compared to range normalization.Clusters in the z-score standardized data plot figure 3.3 seem more pronounced and well-defined, indicating a more effective normalization technique for revealing distinct patterns and structure within the dataset.

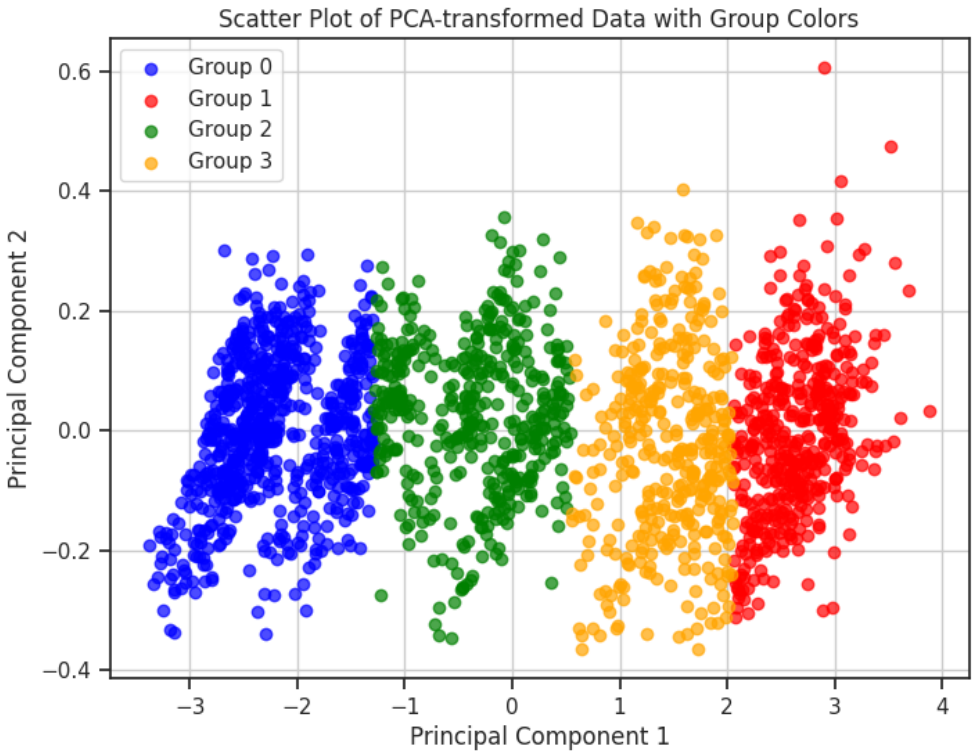


Figure 3.1: scatter plot

There is normalization to scale a ranking factor to a 0-100 range figure 3.4 for ensuring consistent and standardized values across your dataset, allowing for more meaningful comparisons and interpretations.

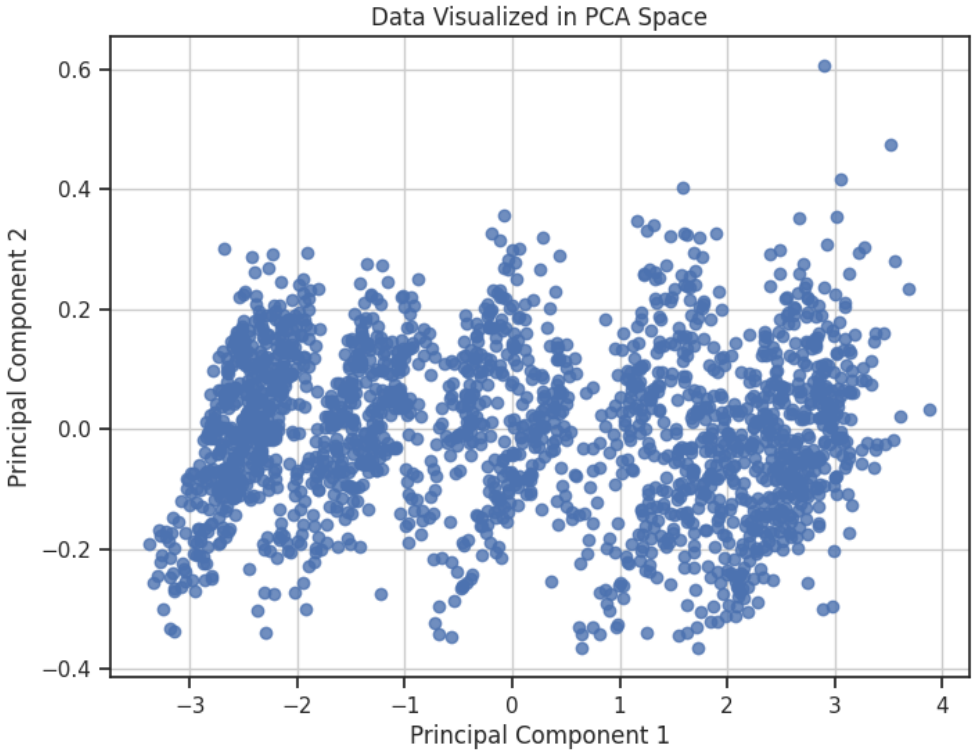


Figure 3.2: scatter plot

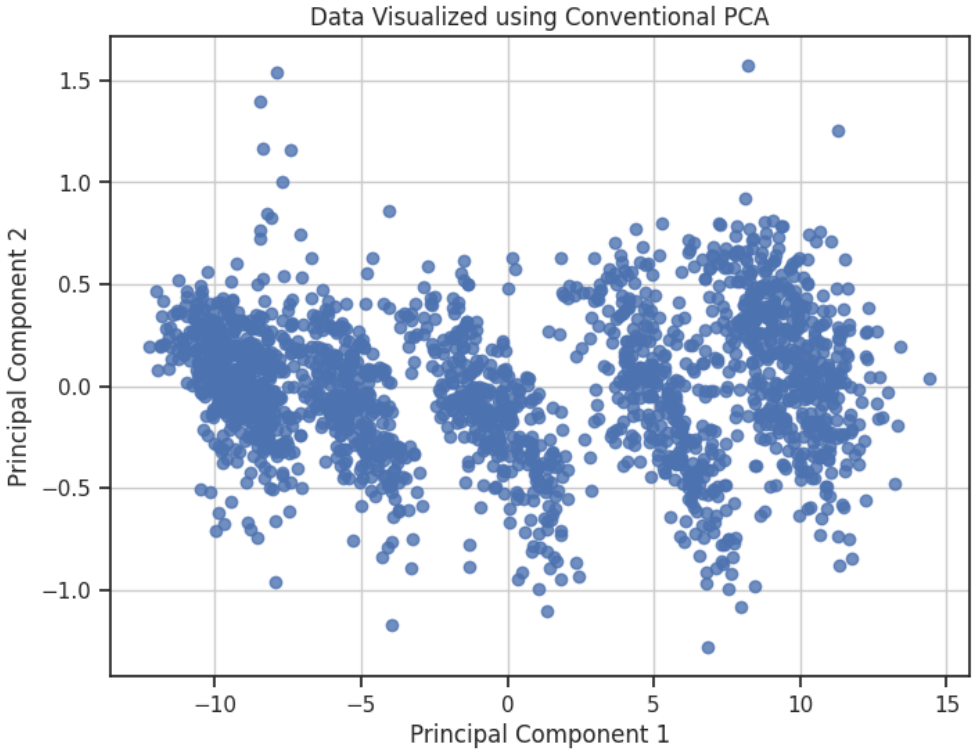


Figure 3.3: scatter plot

Ranking Factor	Normalized Rankings (0-100 scale)
-1.63949145	8.39325981
-1.19947183	20.49673817
-0.95444403	27.2366393
...	...
-0.62382518	36.33086608

Table 3.2: Ranking Factor and Normalized Rankings (0-100 scale)

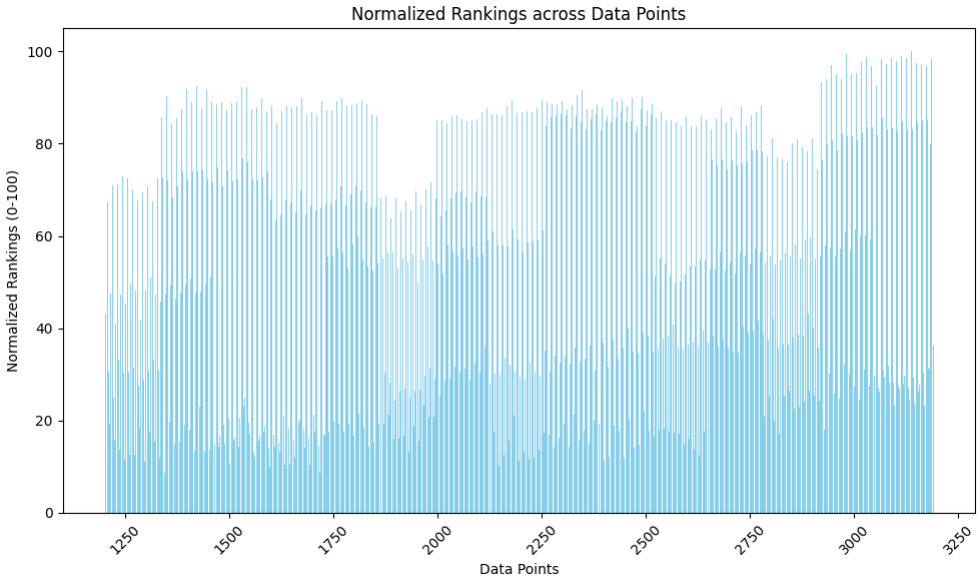


Figure 3.4: Normalized Rankings across Data Points

Chapter 4

Cluster analysis and cluster interpretation

To understand the overall patterns of the courses the subset of the following features was explored via K-means algorithm:

- LandAverageTemperature
- LandMaxTemperature
- LandMinTemperature

The code for that chapter is situated in APPENDIX

Cluster	Relative Difference (Feature 1)	Relative Difference (Feature 2)	Relative Difference
Cluster 1	-0.716	-0.712	-0.719
Cluster 2	0.176	0.195	0.143
Cluster 3	-1.314	-1.325	-1.293
Cluster 4	1.095	1.090	1.103

Table 4.1: Relative Differences for K=4 clusters

Retrieve the cluster centers for each partition generated by K-means clustering (for [**?**] and $K = 7$). These centers are usually stored in variables like centers k4 and centers k7. the grand mean for this dataset for the chosen features. Compute Relative Differences we need to subtract the grand mean from each cluster center to obtain the relative differences. The formula would look like: Relative Difference = Cluster Center Grand Mean

Cluster	Relative Difference (Feature 1)	Relative Difference (Feature 2)	Relative Difference
Cluster 1	0.874	0.867	0.870
Cluster 2	-0.623	-0.626	-0.608
Cluster 3	-1.402	-1.405	-1.392
Cluster 4	-1.033	-1.041	-1.026
Cluster 5	0.028	0.054	0.000
Cluster 6	1.265	1.257	1.280
Cluster 7	0.557	0.566	0.526

Table 4.2: Relative Differences for K=7 clusters

The relative differences between the grand mean and the cluster centers offer valuable insights into the distinctiveness and characteristics of each partition. When evaluating the K=4 partition, it's evident that the clusters showcase notable separation from the grand mean. However, these differences appear to be substantial yet relatively similar across the clusters, potentially indicating a broader categorization of the dataset.

On the contrary, the K=7 partition exhibits more diversified and distinct clusters in terms of their relative differences from the grand mean. The varied distances suggest a finer segmentation of the data, hinting at more specific groupings within the dataset. Notably, some clusters show closer proximity to the grand mean, while others demonstrate more considerable divergence, potentially signifying heterogeneous clusters within this partition.

Considering the balance between complexity and interpretability, the K=4 partition might be more straightforward for interpretation due to its fewer clusters and relatively consistent deviations from the grand mean across these clusters. However, if a deeper understanding of more nuanced groupings within the data is required, the K=7 partition offers a more detailed breakdown, albeit at the cost of increased complexity. The choice between the partitions ultimately depends on the research context and the level of granularity needed for analysis.

Chapter 5

Contingency Table

For studying the conceptual relations, the contingency tables are used.

LandAndOceanAverageTemperature	LandAndOceanAverageTemperatureUncertainty	Occu
12.475	0.299	
12.620	0.187	
12.658	0.378	
12.702	0.186	
12.732	0.372	
...	...	

Table 5.1: Occurrences of Combinations

LandAndOceanAverageTemperature	0.046	0.047	0.048
12.475	**-0.038808**	-0.044811	-0.077615
12.620	**-0.038808**	-0.044811	-0.077615
12.658	**-0.038808**	-0.044811	-0.077615
12.702	**-0.038808**	-0.044811	-0.077615
12.732	**-0.038808**	-0.044811	-0.077615
...	...	...	...

Table 5.2: Adjusted Residuals for LandAndOceanAverageTemperatureUncertainty

LandAndOceanAverageTemperature	0.049	0.050	0.051
12.475	-0.080784	-0.112028	-0.105091
12.620	-0.080784	-0.112028	-0.105091
12.658	-0.080784	-0.112028	-0.105091
12.702	-0.080784	-0.112028	-0.105091
12.732	-0.080784	-0.112028	-0.105091

Table 5.3: Adjusted Residuals for LandAndOceanAverageTemperatureUncertainty

LandAndOceanAverageTemperature	0.052	...	0.378	0.387
12.475	-0.116423	...	-0.031686	**-0.022406**
12.620	-0.116423	...	-0.031686	**-0.022406**
12.658	-0.116423	...	31.527781	**-0.022406**
12.702	-0.116423	...	-0.031686	**-0.022406**
12.732	-0.116423	...	-0.031686	**-0.022406** height

Table 5.4: Adjusted Residuals for LandAndOceanAverageTemperatureUncertainty (Partial)

LandAndOceanAverageTemperatureUncertainty	0.389	0.402	0.414
12.475	-0.022406	-0.022406	-0.022406
12.620	-0.022406	-0.022406	-0.022406
12.658	-0.022406	-0.022406	-0.022406
12.702	-0.022406	-0.022406	-0.022406
12.732	-0.022406	-0.022406	-0.022406 height

Table 5.5: Adjusted Residuals for LandAndOceanAverageTemperatureUncertainty (Partial)

The adjusted residuals (Quetelet Index) represent the deviation of observed frequencies from expected frequencies in a contingency table, indicating the strength and direction of association between categorical variables. In this context, higher absolute values of adjusted residuals suggest stronger associations.

LandAndOceanAverageTemperature	LandAndOceanAverageTemperatureUncertainty				
12.475	-0.022406	-0.022406	-0.038808	-0.044811	-0.077615
12.620	-0.022406	-0.022406	-0.038808	-0.044811	-0.077615
12.658	-0.022406	-0.022406	-0.038808	-0.044811	-0.077615
12.702	-0.022406	-0.022406	-0.038808	-0.044811	-0.077615

Table 5.6: Adjusted Residuals (Quetelet Index)

Looking at the computed adjusted residuals, values deviating significantly from 0 (zero) indicate stronger associations between specific categories of ”LandAndOceanAverageTemperature” and ”LandAndOceanAverageTemperatureUncertainty.” Categories with more extreme values (positive or negative) of adjusted residuals are maximally associated.

For instance, in the presented table, cells with notably high absolute values (far from 0) are indicative of maximally associated categories. These cells represent pairs of categories for which the observed frequencies significantly differ from the expected frequencies, suggesting a stronger association between these specific combinations of ”LandAndOceanAverageTemperature” and ”LandAndOceanAverageTemperatureUncertainty” categories. Therefore, examining cells with the most extreme values

LandAndOceanAverageTemperatureUncertainty	0.417	0.427	0.438
12.475	-0.022406	-0.022406	-0.022406
12.620	-0.022406	-0.022406	-0.022406
12.658	-0.022406	-0.022406	-0.022406
12.702	-0.022406	-0.022406	-0.022406
12.732	-0.022406	-0.022406	-0.022406
...	...	...	...

Table 5.7: Adjusted Residuals for LandAndOceanAverageTemperatureUncertainty (Partial)

LandAndOceanAverageTemperatureUncertainty	0.442	0.457
12.475	-0.022406	-0.022406
12.620	-0.022406	-0.022406
12.658	-0.022406	-0.022406
12.702	-0.022406	-0.022406
12.732	-0.022406	-0.022406

Table 5.8: Adjusted Residuals for LandAndOceanAverageTemperatureUncertainty (Remaining)

of adjusted residuals can help identify maximally associated categories for further analysis or investigation.

High Positive Values: cells in the table of adjusted residuals that have notably high positive values (values substantially greater than 0). These indicate a strong positive association between the corresponding categories of 'LandAndOceanAverageTemperature' and 'LandAndOceanAverageTemperatureUncertainty.'

High Negative Values: cells with notably low (highly negative) values (far less than 0) in the table of adjusted residuals indicate a strong negative association between the respective categories of 'LandAndOceanAverageTemperature' and 'LandAndOceanAverageTemperatureUncertainty.'

by accessing this data here only High Negative Values are present

a high negative value in the cell where 'LandAndOceanAverageTemperature' is 12.475 and 'LandAndOceanAverageTemperatureUncertainty' is 0.046. This would suggest that the occurrence of 'LandAndOceanAverageTemperature' being 12.475 is associated with a significantly lower likelihood of 'LandAndOceanAverageTemperatureUncertainty' being 0.046 than expected by chance.

The Quetelet index (also known as adjusted residuals) measures the devia-

tion of observed frequencies from expected frequencies in a contingency table. average abs residuals equals to 0.10802117764625002 and Chi-squared Statistic: 466644.89099518606.For a contingency table with r rows and c columns, the degrees of freedom (df) can be calculated as$(r - 1)$ * $(c - 1)$. So the average chi-squared value per observation: 0.9945034588920986

Both values being relatively high indicate a considerable discrepancy between observed and expected frequencies in the contingency table.This suggests that the fit between the observed and expected values might not be as strong or that there might be some relationship between the variables not accounted for in the model.

Critical value for 95 percent confidence level: 468926.35327760974 Critical value for 99 percent confidence level: 469587.01517297537 This suggests that there's no statistically significant association at either of the confidence level. In other words, based on the chi-squared test, there's insufficient evidence to conclude that the variables are associated with each other.

Chapter 6

Bootstrap

Bootstrapping is used for estimating of a sampling distribution of various statistics.95 percent confidence interval can be found after calculating estimations of the mean and standard deviation of the bootstrapped means.

Grand Mean Comparison:

Method	**Confidence Interval**
Pivotal	$[-2.75 \times 10^{-17}, 3.35 \times 10^{-17}]$
Non-pivotal	$[-6.41 \times 10^{-17}, 6.41 \times 10^{-17}]$

Table 6.1: Comparison of Confidence Intervals using Pivotal and Non-pivotal Methods

This interval contains zero, indicating that the grand mean is not significantly different from zero (considering the precision).Similar to the pivotal method, this interval contains zero, indicating that the grand mean is not significantly different from zero. Both pivotal and non-pivotal methods suggest that the grand mean is not significantly different from zero.

Within-Cluster Mean Comparison in Cluster 1:

Method	**Confidence Interval**
Non-pivotal	[1.457, 1.764]
Pivotal	[1.460, 1.770]

Table 6.2: Comparison of Confidence Intervals for Within-Cluster Mean in Cluster 1

The confidence interval does not contain the grand mean of 1.61.Similar to the pivotal method, the grand mean of 1.61 falls outside this interval. Both pivotal and non-pivotal methods suggest that the grand mean of the entire dataset (including both

clusters) is significantly different from the within-cluster mean in Cluster 1 for the specific feature. The grand mean is not within the confidence intervals obtained for the within-cluster mean in Cluster 1.

In summary, while the grand mean of the entire dataset does not significantly differ from zero, it significantly differs from the within-cluster mean in Cluster 1 for the specific feature. This indicates that the mean within Cluster 1 for that feature is different from the overall dataset's grand mean.

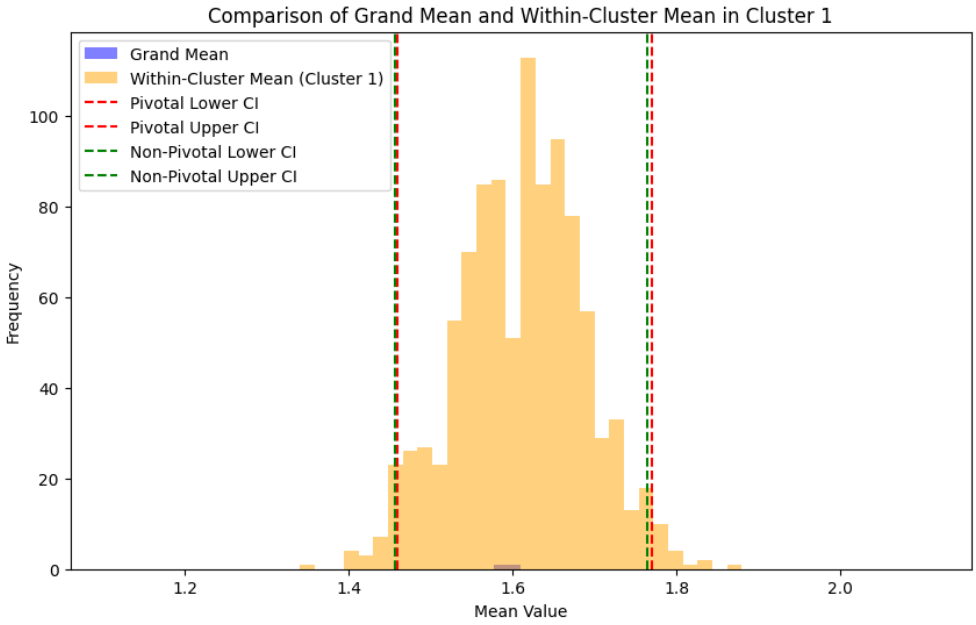


Figure 6.1: bootstrap

Chapter 7

REFERENCES

1. https://www.kaggle.com/datasets/berkeleyearth/climate-change-earth-surface-temperature-data/data?select=GlobalTemperatures.csv

2. The Elements of Statistical Learning Data Mining, Inference, and Prediction, Second Edition Springer (2019).

3. Mirkin., B., Core Data Analysis: Summarization, Correlation, And Visualization. 2nd ed. London: Springer-Verlag London Limited (2019).

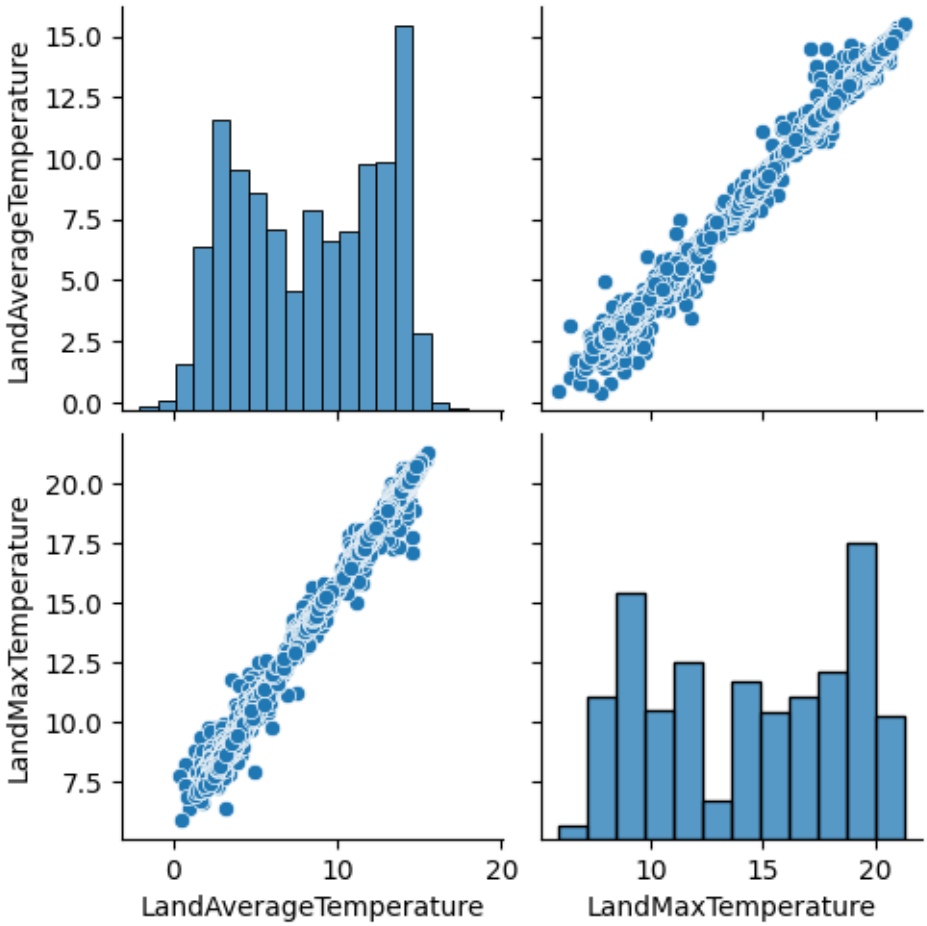
Chapter 8

APPENDIX

da-1

December 15, 2023

[8]:
```
#Find two feature in my dataset.
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_csv('/content/GlobalTemperatures.csv')

# Select a subset of features
selected_features = [
    'LandAverageTemperature',
    'LandMaxTemperature',

]

# Pairplot to visualize relationships
sns.pairplot(data[selected_features], kind='scatter')
plt.show()
```



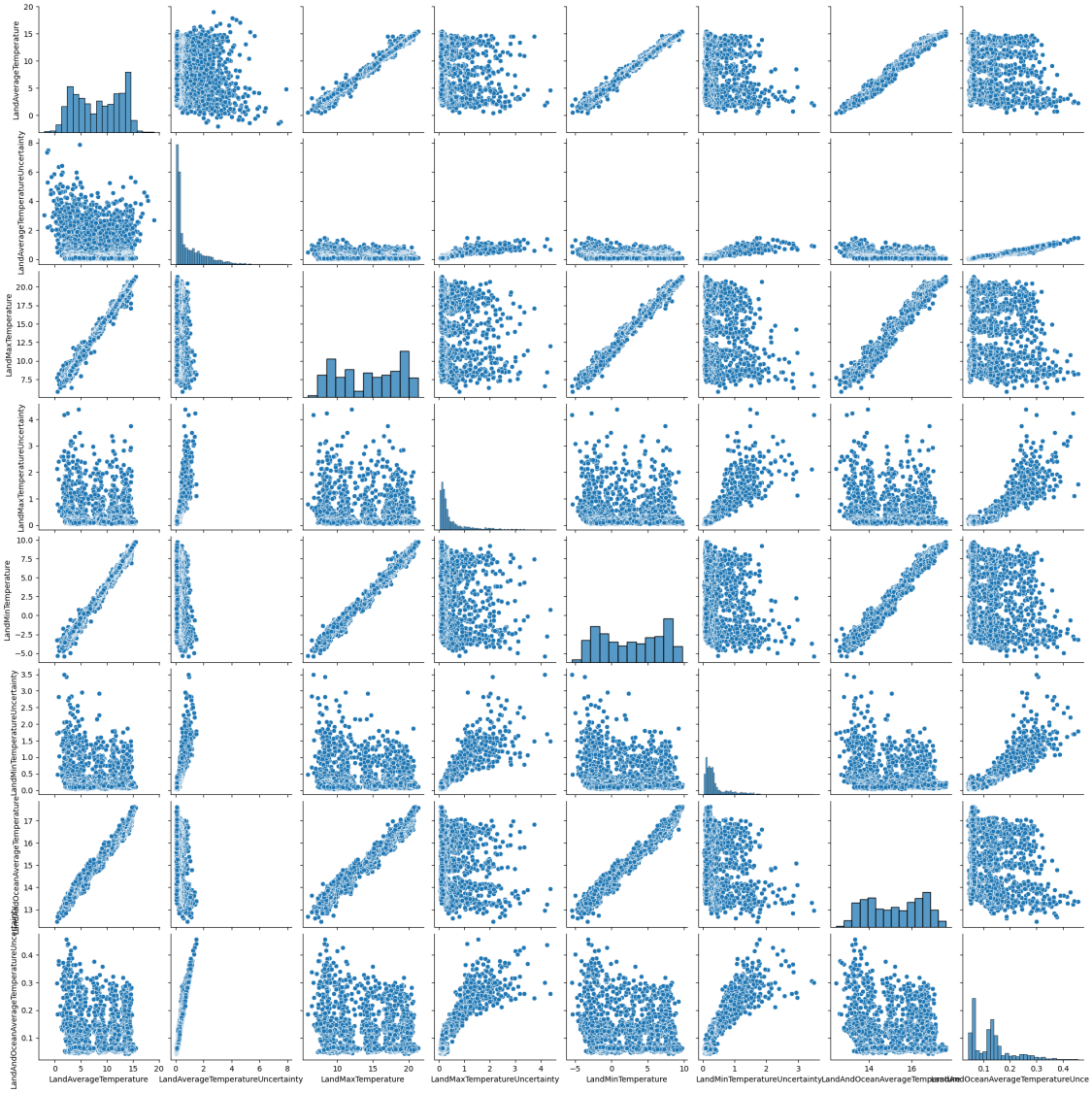
[62]:
```
import pandas as pd

# Drop rows with NaN values
data_without_nan = data.dropna()

# Print the resulting table
print(data_without_nan)
```

```
             dt  LandAverageTemperature  LandAverageTemperatureUncertainty  \
1200  1850-01-01                   0.749                              1.105
1201  1850-02-01                   3.071                              1.275
1202  1850-03-01                   4.954                              0.955
1203  1850-04-01                   7.217                              0.665
1204  1850-05-01                  10.004                              0.617
...          ...                     ...                                ...
3187  2015-08-01                  14.755                              0.072
3188  2015-09-01                  12.999                              0.079
3189  2015-10-01                  10.801                              0.102
```

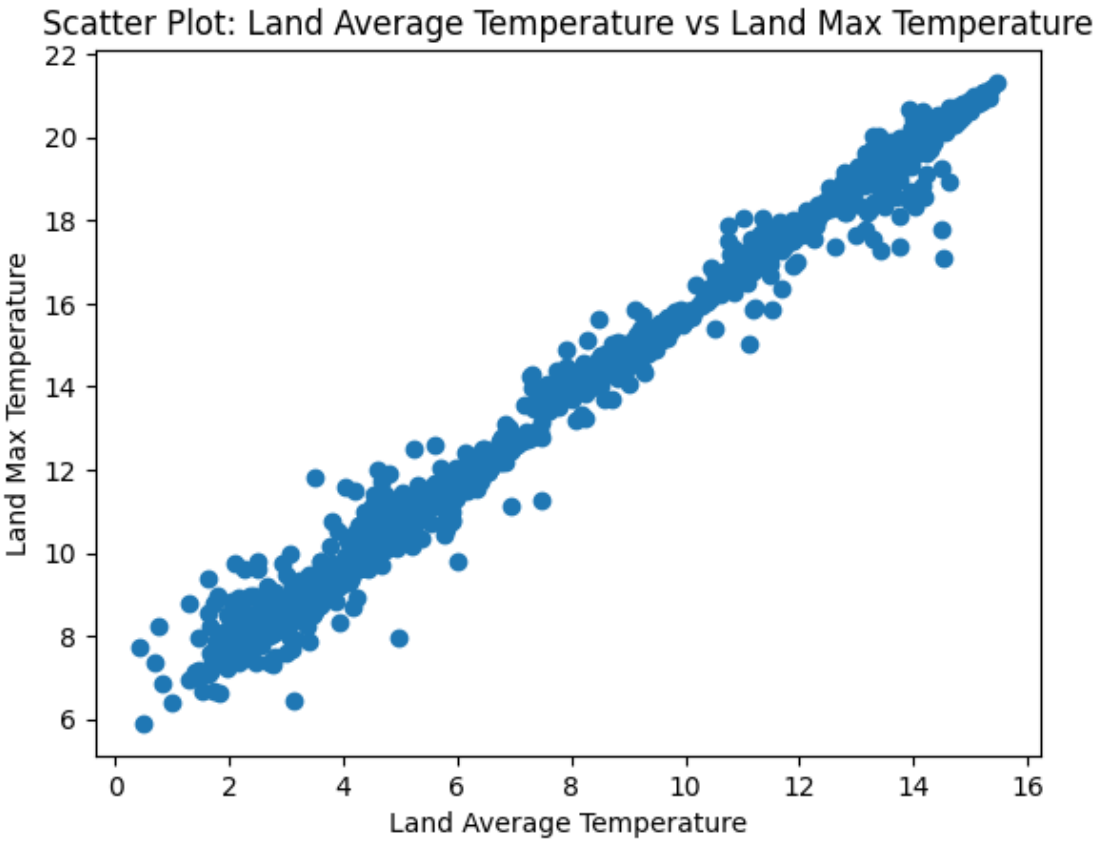
```
3190  2015-11-01                   7.433                                0.119
3191  2015-12-01                   5.518                                0.100

      LandMaxTemperature  LandMaxTemperatureUncertainty  LandMinTemperature  \
1200               8.242                          1.738              -3.206
1201               9.970                          3.007              -2.291
1202              10.347                          2.401              -1.905
1203              12.934                          1.004               1.018
1204              15.655                          2.406               3.811
…                    …                              …                   …
3187              20.699                          0.110               9.005
3188              18.845                          0.088               7.199
3189              16.450                          0.059               5.232
3190              12.892                          0.093               2.157
3191              10.725                          0.154               0.287

      LandMinTemperatureUncertainty  LandAndOceanAverageTemperature  \
1200                          2.822                          12.833
1201                          1.623                          13.588
1202                          1.410                          14.043
1203                          1.329                          14.667
1204                          1.347                          15.507
…                                 …                               …
3187                          0.170                          17.589
3188                          0.229                          17.049
3189                          0.115                          16.290
3190                          0.106                          15.252
3191                          0.099                          14.774

      LandAndOceanAverageTemperatureUncertainty
1200                                      0.367
1201                                      0.414
1202                                      0.341
1203                                      0.267
1204                                      0.249
…                                             …
3187                                      0.057
3188                                      0.058
3189                                      0.062
3190                                      0.063
3191                                      0.062

[1992 rows x 9 columns]
```

[63]:
```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
```

```
# Select specific columns
selected_columns = ['dt', 'LandAverageTemperature',␣
 ↪'LandAverageTemperatureUncertainty',
                    'LandMaxTemperature', 'LandMaxTemperatureUncertainty',
                    'LandMinTemperature', 'LandMinTemperatureUncertainty',
                    'LandAndOceanAverageTemperature',
                    'LandAndOceanAverageTemperatureUncertainty']

# Subset the DataFrame with selected columns
selected_data = data[selected_columns]

# Create pairwise scatterplots
sns.pairplot(selected_data)
plt.show()
```



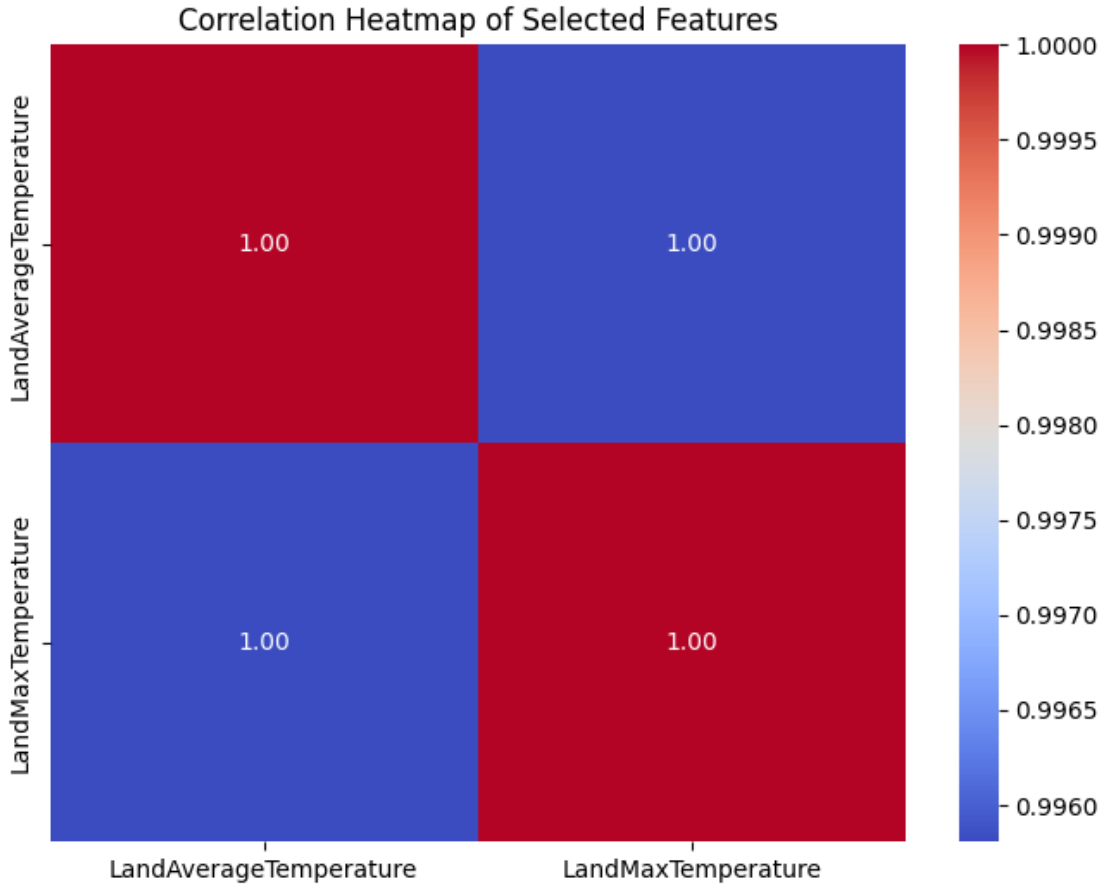
[11]:
```
import matplotlib.pyplot as plt

plt.scatter(data['LandAverageTemperature'], data['LandMaxTemperature'])
plt.xlabel('Land Average Temperature')
plt.ylabel('Land Max Temperature')
plt.title('Scatter Plot: Land Average Temperature vs Land Max Temperature')
plt.show()
```



[12]:
```
# Extract the selected features
selected_data = data[selected_features]

# Calculate the correlation matrix
correlation_matrix = selected_data.corr()

# Display the correlation matrix
print(correlation_matrix)

# Create a heatmap of the correlation matrix
```

```
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap of Selected Features')
plt.show()
```

	LandAverageTemperature	LandMaxTemperature
LandAverageTemperature	1.000000	0.995807
LandMaxTemperature	0.995807	1.000000



[13]:
```
from sklearn.linear_model import LinearRegression
from sklearn.impute import SimpleImputer

# Features and target variable
features = ['LandAverageTemperature']
target = 'LandMaxTemperature'

# Extract features and target variable
X = data[features]
y = data[target]
```

```
# Initialize the SimpleImputer and fit it to fill missing values with the mean
imputer = SimpleImputer(strategy='mean')
X_imputed = imputer.fit_transform(X)

# Initialize the SimpleImputer and fit it to fill missing values with the mean
imputer = SimpleImputer(strategy='mean')
y_imputed = imputer.fit_transform(y.values.reshape(-1, 1)).ravel()

# Initialize the Linear Regression model
model = LinearRegression()

# Fit the model with imputed data
model.fit(X_imputed, y_imputed)

# Get the coefficients (slope) and intercept
coefficients = model.coef_
intercept = model.intercept_

print(f"Coefficients: {coefficients}")
print(f"Intercept: {intercept}")
# Get the coefficients (slopes) for each predictor variable
slopes = model.coef_

# Print the slopes for each predictor variable
for i, slope in enumerate(slopes):
    print(f"Slope for Predictor {i+1}: {slope}")
```

```
Coefficients: [0.59692238]
Intercept: 9.351536463007708
Slope for Predictor 1: 0.5969223801657567
```

[14]:
```
# Make predictions on the test set
da_predictions = model.predict(X_imputed)

# Make predictions on the test set
ml_predictions = model.predict(X_imputed)


true_values = y_imputed
```

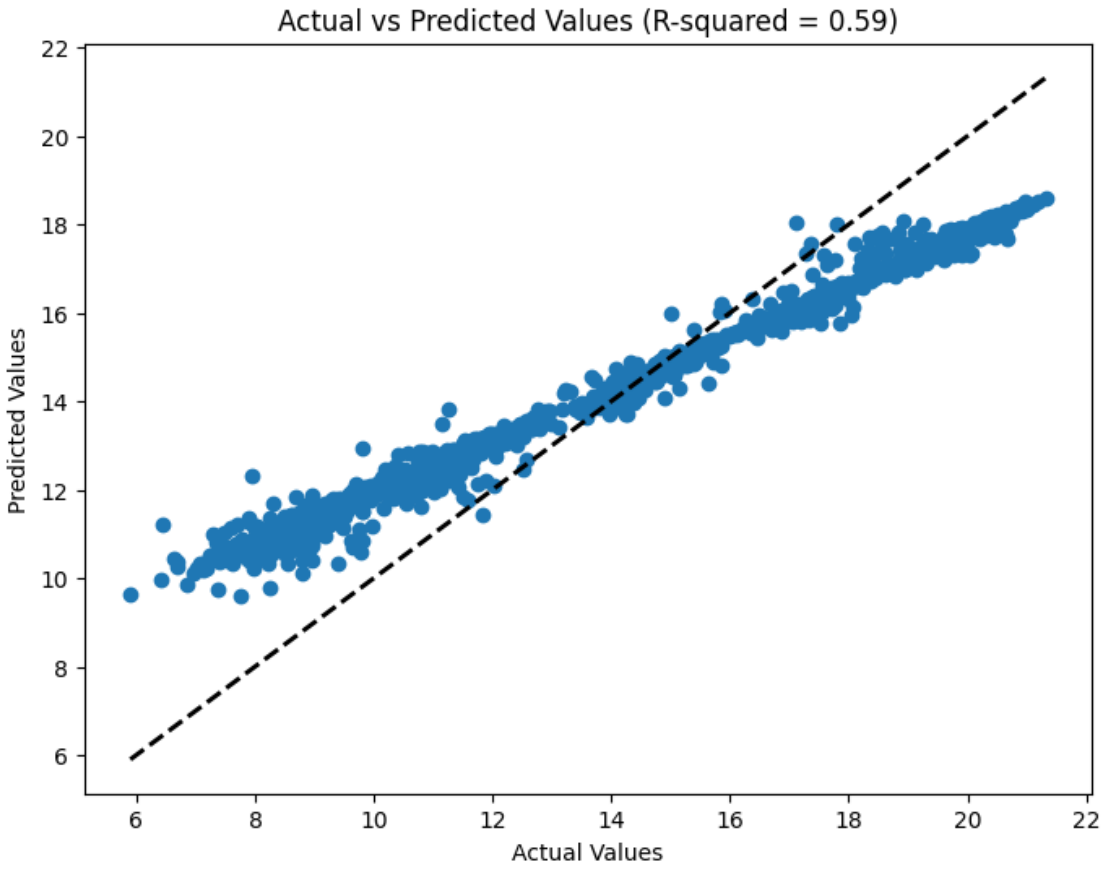
[15]:
```
import pandas as pd


# Select the columns for which you want to find correlations
selected_columns = ['LandAverageTemperature', 'LandMaxTemperature']
```

```
# Create a subset DataFrame with the selected columns
subset_data = data[selected_columns]

# Calculate the correlation matrix
correlation_matrix = subset_data.corr()

print(correlation_matrix)
```

```
                         LandAverageTemperature  LandMaxTemperature
LandAverageTemperature                 1.000000            0.995807
LandMaxTemperature                     0.995807            1.000000
```

[16]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score


# and have X (features) and y (target variable)

# Fit the model
model.fit(X_imputed, y_imputed)

# Make predictions
predictions = model.predict(X_imputed)

# Calculate R-squared
r_squared = r2_score(y_imputed, predictions)
print("Coefficient of determination (R-squared):", r_squared)

# Plotting actual vs predicted
plt.figure(figsize=(8, 6))
plt.scatter(y, predictions)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs Predicted Values (R-squared = {:.2f})'.format(r_squared))
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw=2)  # Diagonal line␣
 ↪for reference
plt.show()
```

```
Coefficient of determination (R-squared): 0.5880215281554846
```



[17]:
```
X = data[['LandAverageTemperature']]  # Adjust this according to your model's input features

# Perform predictions using the model
```

[18]:
```
from sklearn.metrics import r2_score
import numpy as np

# Replace y_true and y_pred with your actual and predicted values
y_true = y_imputed
  # Actual target values
y_pred = model.predict(X_imputed) # Predicted values from your model

# Calculate the correlation coefficient
correlation = np.corrcoef(y_true, y_pred)[0, 1]

# Calculate the determinacy coefficient (R^2)
determinacy = r2_score(y_true, y_pred)
```

```
print(f"Correlation coefficient: {correlation}")
print(f"Determinacy coefficient (R^2): {determinacy}")
```

```
Correlation coefficient: 0.7668256178268203
Determinacy coefficient (R^2): 0.5880215281554846
```

[19]:
```
new_predictors = [[0.749  ], [3.071 ], [4.954 ], [7.217]]  # Example predictor␣
↪values

# Make predictions
predictions = model.predict(new_predictors)

# Display predicted LandMaxTemperature values
for predictor, prediction in zip(new_predictors, predictions):
    print(f"Predictor value: {predictor}, Predicted LandMaxTemperature:␣
↪{prediction}")

    #8.242  9.970 10.347 12.934
```

```
Predictor value: [0.749], Predicted LandMaxTemperature: 9.79863132575186
Predictor value: [3.071], Predicted LandMaxTemperature: 11.184685092496746
Predictor value: [4.954], Predicted LandMaxTemperature: 12.308689934348866
Predictor value: [7.217], Predicted LandMaxTemperature: 13.659525280663974
```

[20]:
```
# Example coefficients and intercept
m1 = 0.59692238
b = 9.351536463007708

# Example predictor values
predictor1 = 0.5969223801657567


# Calculate LandMaxTemperature using the formula
land_max_temperature = m1 * predictor1 + b
print(f"Predicted LandMaxTemperature: {land_max_temperature}")
```

```
Predicted LandMaxTemperature: 9.707852790851517
```

[21]:
```
import numpy as np


# Calculate Mean Relative Absolute Error for DA approach
da_errors = np.abs(da_predictions - true_values) / true_values
da_mrae = da_errors.mean()

# Calculate Mean Relative Absolute Error for ML approach
ml_errors = np.abs(ml_predictions - true_values) / true_values
```

```
ml_mrae = ml_errors.mean()

print(f"Mean Relative Absolute Error for DA approach: {da_mrae}")
print(f"Mean Relative Absolute Error for ML approach: {ml_mrae}")
```

```
Mean Relative Absolute Error for DA approach: 0.14417666757165506
Mean Relative Absolute Error for ML approach: 0.14417666757165506
```

[24]:
```
import pandas as pd
from sklearn.preprocessing import StandardScaler

selected_features = data[['LandAverageTemperature', 'LandMaxTemperature',␣
 ↪'LandMinTemperature', 'LandAndOceanAverageTemperature']]  # Replace with␣
 ↪your selected features
# Drop rows with missing values in the selected features
cleaned_data = selected_features.dropna()
scaler = StandardScaler()
scaled_data = scaler.fit_transform(cleaned_data)
```

[25]:
```
# Compute the scatter matrix
scatter_matrix = np.cov(scaled_data.T)
```

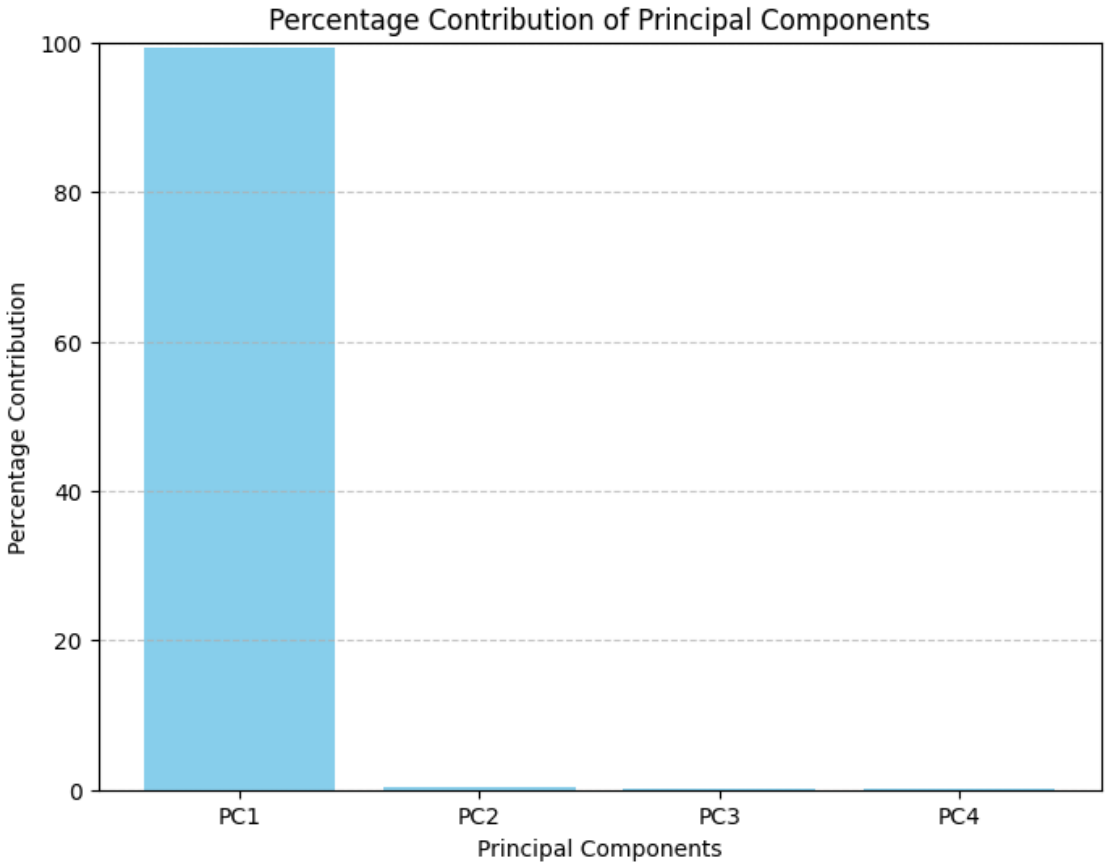
[26]:
```
from numpy.linalg import svd

# Perform SVD
U, s, Vt = svd(scaled_data)
```

[27]:
```
# Calculate percentage contribution of each principal component
explained_variance_ratio = (s ** 2) / np.sum(s ** 2)
percent_contributions = explained_variance_ratio * 100

print("Percentage Contribution of Principal Components:")
for i, percent in enumerate(percent_contributions):
    print(f"PC{i+1}: {percent:.2f}%")
```

```
Percentage Contribution of Principal Components:
PC1: 99.29%
PC2: 0.47%
PC3: 0.16%
PC4: 0.08%
```

[28]:
```
import matplotlib.pyplot as plt

# Principal component labels
components = ['PC1', 'PC2', 'PC3', 'PC4']

# Percentage contribution values
```

```
contributions = [99.29, 0.47, 0.16, 0.08]

# Create bar plot
plt.figure(figsize=(8, 6))
plt.bar(components, contributions, color='skyblue')
plt.xlabel('Principal Components')
plt.ylabel('Percentage Contribution')
plt.title('Percentage Contribution of Principal Components')
plt.ylim(0, 100)  # Set y-axis limit to percentage range
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



[29]:
```
import pandas as pd

# Calculate the minimum and maximum values for the selected columns
min_vals = cleaned_data.min()
max_vals = cleaned_data.max()

# Perform range normalization for the selected columns
```

```
range_normalized_data = (cleaned_data - min_vals) / (max_vals - min_vals)

# 'range_normalized_data' now holds the range-normalized values for the␣
 ↪selected columns
print(f"The range of the dataset is: {range_normalized_data}")
```

```
The range of the dataset is:       LandAverageTemperature  LandMaxTemperature
LandMinTemperature  \
1200                0.022881            0.151881            0.145550
1201                0.176880            0.263943            0.206057
1202                0.301764            0.288392            0.231583
1203                0.451850            0.456161            0.424878
1204                0.636689            0.632620            0.609575
…                        …                   …                   …
3187                0.951784            0.959728            0.953049
3188                0.835323            0.839494            0.833620
3189                0.689548            0.684176            0.703545
3190                0.466176            0.453437            0.500198
3191                0.339170            0.312905            0.376537

      LandAndOceanAverageTemperature
1200                        0.069704
1201                        0.216706
1202                        0.305296
1203                        0.426791
1204                        0.590343
…                                …
3187                        0.995717
3188                        0.890576
3189                        0.742796
3190                        0.540693
3191                        0.447625

[1992 rows x 4 columns]
```

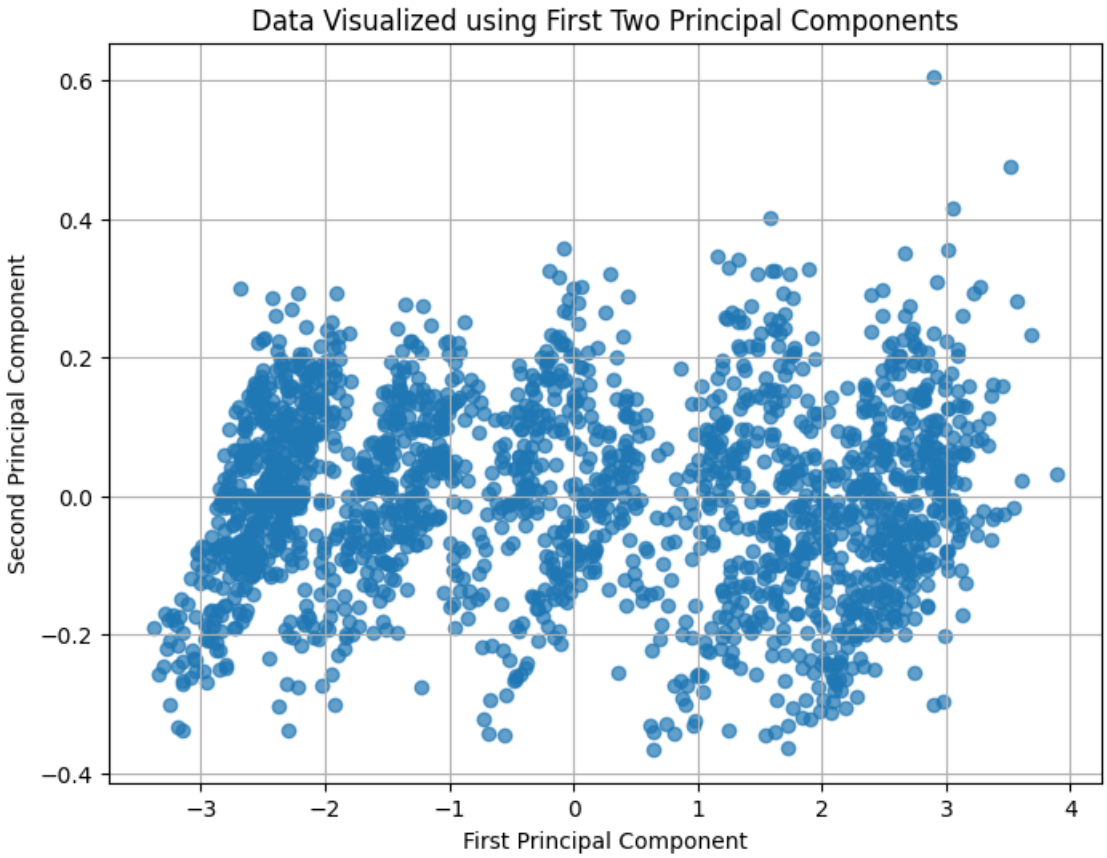
[30]:
```
from sklearn.decomposition import PCA
import numpy as np


# It should be a NumPy array or DataFrame containing your preprocessed data

# Initialize PCA with 2 components for visualization
pca = PCA(n_components=2)

# Fit PCA on the range-normalized data
pca.fit(range_normalized_data)
```

```
# Transform the data into the principal components
principal_components = pca.transform(range_normalized_data)

# 'principal_components' now holds the transformed data with reduced dimensions
# You can further use this for visualization or analysis
```

[31]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Standardize the data
scaler = StandardScaler()
standardized_data = scaler.fit_transform(cleaned_data)

# Apply PCA with two components
pca = PCA(n_components=2)
pca_data = pca.fit_transform(standardized_data)

# 'pca_data' now contains the PCA-transformed data with two components
print(f"The pca_data of the dataset is: {pca_data }")
```

```
The pca_data of the dataset is: [[ 3.27630426  0.3023267 ]
 [ 2.39716075  0.12406358]
 [ 1.90755162 -0.03379915]
 …
 [-1.22705203 -0.27525945]
 [ 0.35840464 -0.25442974]
 [ 1.24734352 -0.33863221]]
```

[32]:
```
import matplotlib.pyplot as plt

# Plotting the first two principal components
plt.figure(figsize=(8, 6))
plt.scatter(pca_data[:, 0], pca_data[:, 1], alpha=0.7)
plt.xlabel('First Principal Component')
plt.ylabel('Second Principal Component')
plt.title('Data Visualized using First Two Principal Components')
plt.grid(True)
plt.show()
```

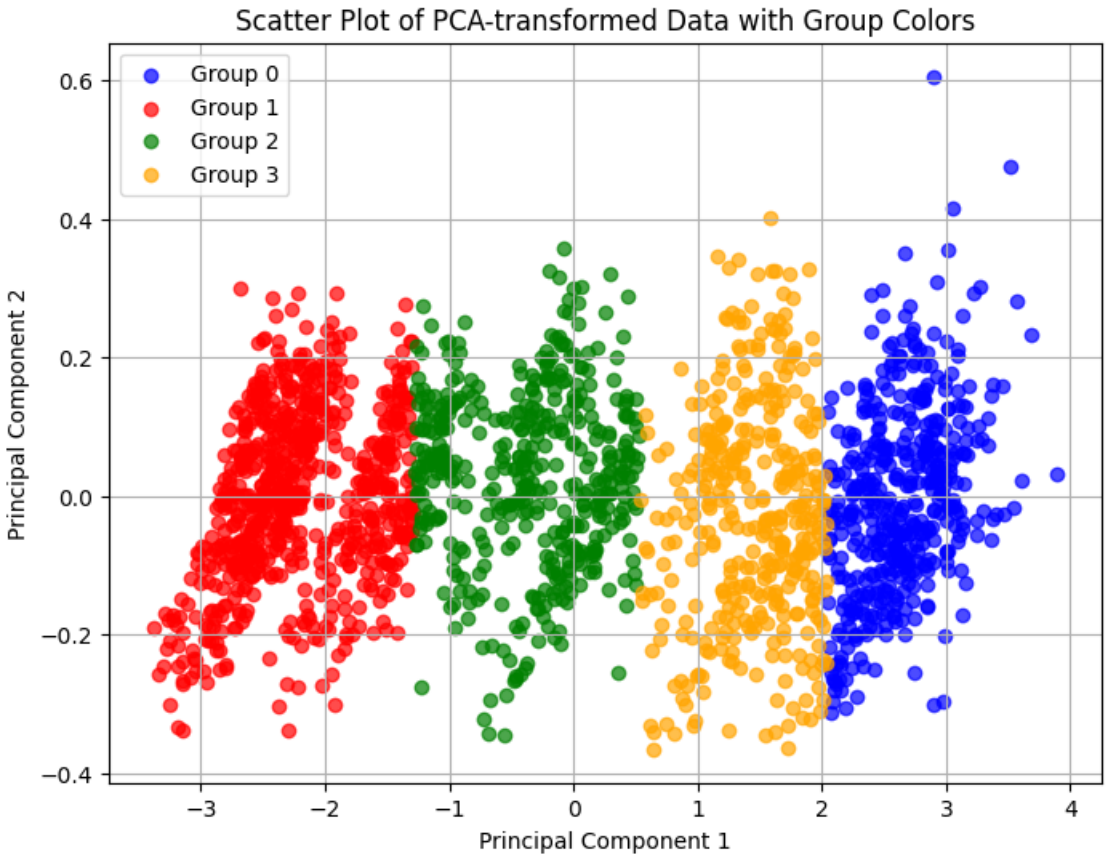


[33]:
```
from sklearn.cluster import KMeans

# You can perform K-means clustering to generate labels
kmeans = KMeans(n_clusters=4)  # Specify the number of clusters
labels = kmeans.fit_predict(pca_data)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```

[34]:
```
import matplotlib.pyplot as plt
import numpy as np


unique_labels = np.unique(labels)

# Define a color map or a list of colors for each unique group
colors = ['blue', 'red', 'green', 'orange']  # Add more colors as needed
```

```
plt.figure(figsize=(8, 6))

# Scatter plot with distinct colors for each group
for i, label in enumerate(unique_labels):
    indices = np.where(labels == label)
    plt.scatter(
        pca_data[indices, 0],
        pca_data[indices, 1],
        label=f'Group {label}',
        alpha=0.7,
        color=colors[i],
    )

plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('Scatter Plot of PCA-transformed Data with Group Colors')
plt.legend()
plt.grid(True)
plt.show()
```

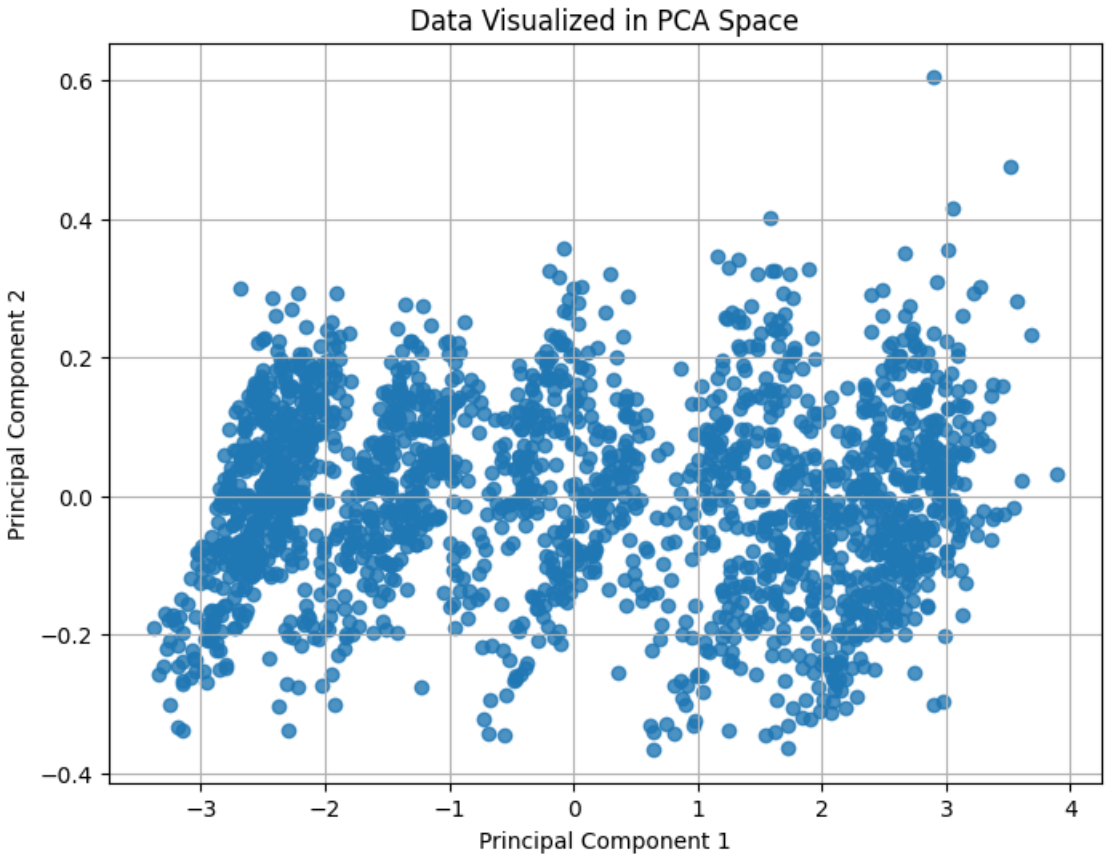


[35]:
```
from sklearn.preprocessing import StandardScaler

# Assuming 'cleaned_data' contains your data without NaN values
scaler = StandardScaler()
z_score_standardized_data = scaler.fit_transform(cleaned_data)
print(f"The Z_score of the dataset is: {z_score_standardized_data}")
```

```
The Z_score of the dataset is: [[-1.83537261 -1.41780318 -1.43198391
-1.86812372]
 [-1.29057362 -1.01673525 -1.21175628 -1.2753965 ]
 [-0.84877496 -0.92923375 -1.11885151 -0.91819002]
 …
 [ 0.52307675  0.48726947  0.59892399  0.8458604 ]
 [-0.26713987 -0.33854053 -0.14118526  0.0309586 ]
 [-0.71644653 -0.84150014 -0.59126795 -0.34430446]]
```

[36]:
```
from sklearn.decomposition import PCA

# Perform PCA with two components
pca = PCA(n_components=2)
pca.fit(z_score_standardized_data)

# Transform the data into the PCA space
pca_transformed_data = pca.transform(z_score_standardized_data)
```

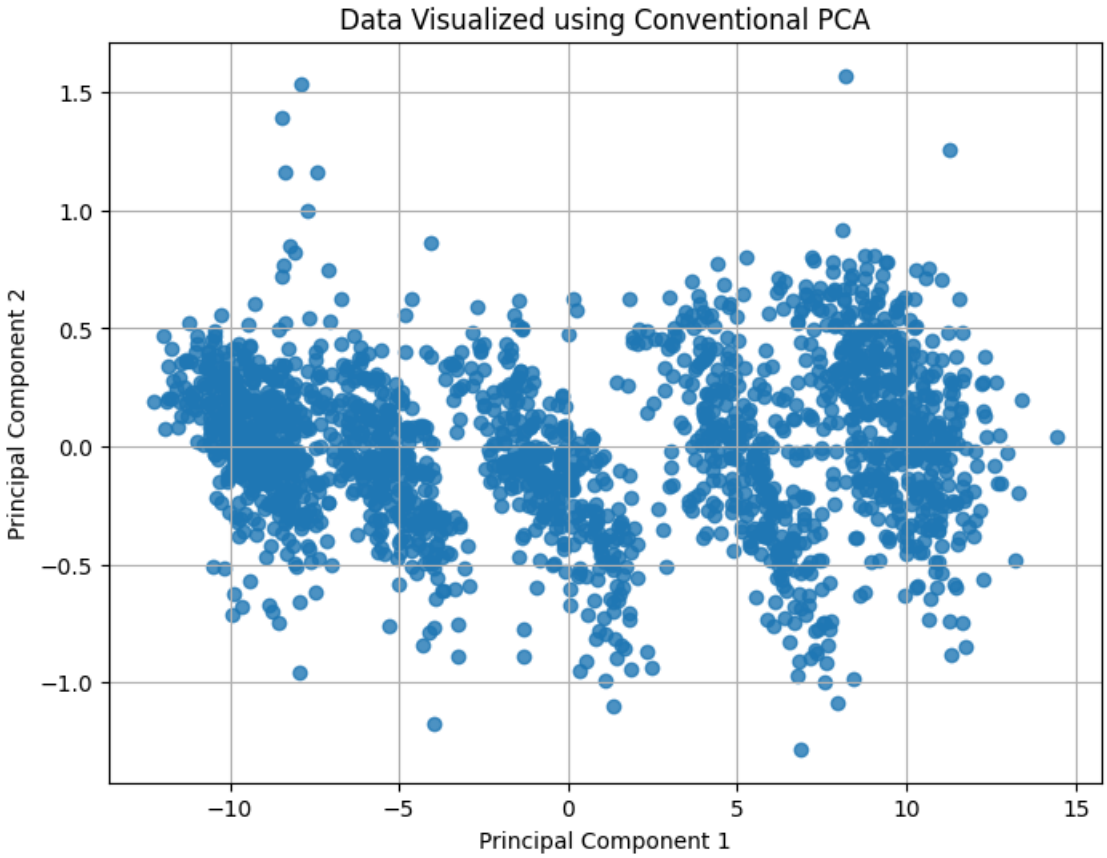
[37]:
```
import matplotlib.pyplot as plt


plt.figure(figsize=(8, 6))

# Scatter plot using the first two principal components
plt.scatter(pca_transformed_data[:, 0], pca_transformed_data[:, 1], alpha=0.8)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('Data Visualized in PCA Space')

plt.grid()
plt.show()
```



[38]:
```python
from sklearn.decomposition import PCA

# Initialize PCA
pca = PCA(n_components=2)  # Specify the number of components

# Fit PCA to the cleaned_data
pca.fit(cleaned_data)

# Transform the data into the PCA space
pca_transformed_data = pca.transform(cleaned_data)

print(f"The pcs transformed of the dataset is: {pca_transformed_data }")

# Plotting the transformed data
plt.figure(figsize=(8, 6))
plt.scatter(pca_transformed_data[:, 0], pca_transformed_data[:, 1], alpha=0.8)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('Data Visualized using Conventional PCA')
plt.grid()
```

```
plt.show()
```

```
The pcs transformed of the dataset is: [[11.72107284 -0.13878298]
 [ 8.75699093 -0.61644776]
 [ 7.17006012 -0.53830979]
 …
 [-4.05678655  0.38111135]
 [ 1.81415579  0.62827019]
 [ 5.28423896  0.80227912]]
```



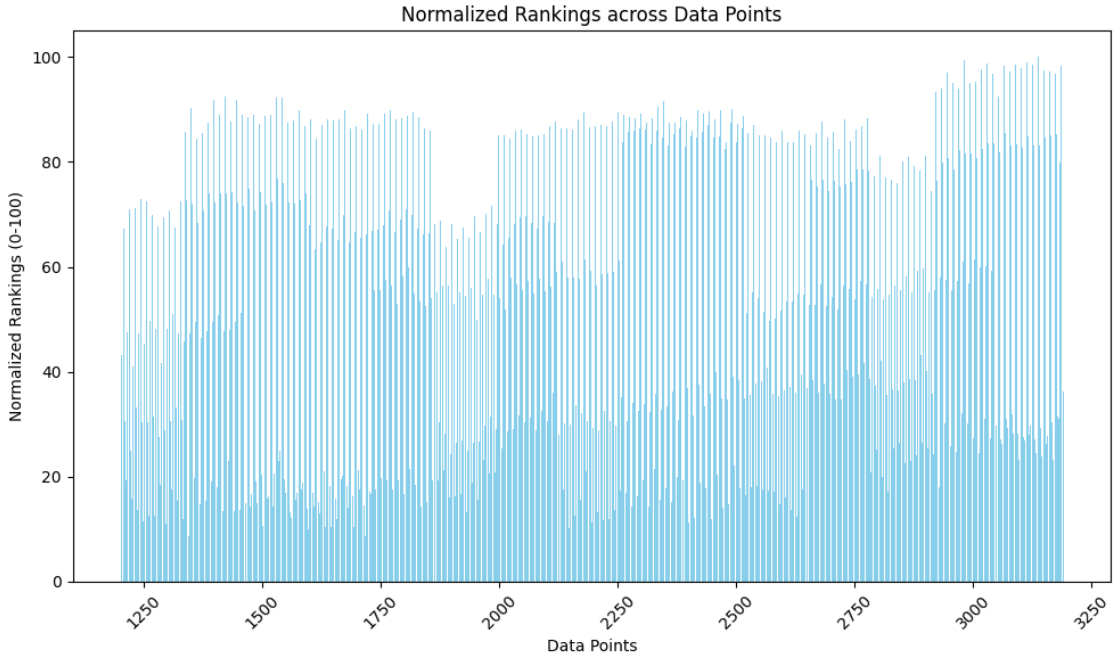
[40]:
```
scaler = StandardScaler()
scaled_data = scaler.fit_transform(cleaned_data)

# Calculate the mean or median across the features for each row
ranking_factor = scaled_data.mean(axis=1)  # You can use .median() instead of .
 ↪mean() if preferred

# Normalize the ranking factor to a 0-100 scale
normalized_ranking = (ranking_factor - ranking_factor.min()) / (ranking_factor.
 ↪max() - ranking_factor.min()) * 100
```

```
# Assign the normalized ranking to a new column in the DataFrame using .loc
cleaned_data.loc[:, 'Ranking'] = normalized_ranking

# Print the ranking factor and the normalized rankings
print("Ranking Factor:")
print(ranking_factor)
print("\nNormalized Rankings (0-100 scale):")
print(normalized_ranking)
```

```
Ranking Factor:
[-1.63949145 -1.19947183 -0.95444403 …  0.6142212  -0.17910464
 -0.62382518]

Normalized Rankings (0-100 scale):
[ 8.39325981 20.49673817 27.23663933 … 70.38541072 48.56365114
 36.33086608]
```

```
<ipython-input-40-05ba3726a272>:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  cleaned_data.loc[:, 'Ranking'] = normalized_ranking
```

[41]:
```
import matplotlib.pyplot as plt


plt.figure(figsize=(10, 6))
plt.bar(cleaned_data.index, normalized_ranking, color='skyblue')
plt.xlabel('Data Points')
plt.ylabel('Normalized Rankings (0-100)')
plt.title('Normalized Rankings across Data Points')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



[42]:
```
import pandas as pd
from sklearn.preprocessing import StandardScaler

selected_features = data[['LandAverageTemperature', 'LandMaxTemperature',␣
 ↪'LandMinTemperature']]  # Replace with your selected features
# Drop rows with missing values in the selected features
cleaned_data = selected_features.dropna()
scaler = StandardScaler()
scaled_data = scaler.fit_transform(cleaned_data)
```

[51]:
```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import numpy as np



selected_features = data[['LandAverageTemperature', 'LandMaxTemperature',␣
 ↪'LandMinTemperature']]  # Replace with your selected features
# Drop rows with missing values in the selected features
cleaned_data = selected_features.dropna()
scaler = StandardScaler()
scaled_data = scaler.fit_transform(cleaned_data)

# Perform K-means clustering for K=4 and K=7
```

```
k_values = [4, 7]
results = {}

for k in k_values:
    cluster_results = []
    for i in range(10):
        # Run K-means clustering
        kmeans = KMeans(n_clusters=k, init='random', n_init=1)
        kmeans.fit(scaled_data)
        cluster_results.append(kmeans.inertia_)  # Recording the criterion value
    results[k] = cluster_results

# Choosing the best runs for K=4 and K=7
best_runs = {k: np.min(v) for k, v in results.items()}
best_k4_run = np.argmin(results[4])
best_k7_run = np.argmin(results[7])

# Retrieve cluster centers for the best runs
kmeans_best_k4 = KMeans(n_clusters=4, init='random', n_init=1,␣
 ↪random_state=best_k4_run)
kmeans_best_k4.fit(scaled_data)
centers_k4 = kmeans_best_k4.cluster_centers_

kmeans_best_k7 = KMeans(n_clusters=7, init='random', n_init=1,␣
 ↪random_state=best_k7_run)
kmeans_best_k7.fit(scaled_data)
centers_k7 = kmeans_best_k7.cluster_centers_

# Compute the grand mean of the standardized data
grand_mean = scaled_data.mean(axis=0)

# Calculate relative differences for K=4 clusters
relative_diff_k4 = centers_k4 - grand_mean

# Calculate relative differences for K=7 clusters
relative_diff_k7 = centers_k7 - grand_mean

# Display or print the results
print("Grand Mean for Selected Features (Standardized):")
print(grand_mean)

print("\nRelative Differences for K=4 clusters:")
print(relative_diff_k4)

print("\nRelative Differences for K=7 clusters:")
print(relative_diff_k7)
```

```
Grand Mean for Selected Features (Standardized):
[-2.85358528e-17  3.56698161e-17 -7.13396321e-18]

Relative Differences for K=4 clusters:
[[-0.7160032  -0.71210146 -0.71859613]
 [ 0.17623287  0.1953575   0.14324563]
 [-1.31373523 -1.32509003 -1.29323128]
 [ 1.09534413  1.08959869  1.10262087]]

Relative Differences for K=7 clusters:
[[ 8.74080842e-01  8.66729244e-01  8.70335690e-01]
 [-6.22759231e-01 -6.25587144e-01 -6.08455757e-01]
 [-1.40178656e+00 -1.40495085e+00 -1.39199296e+00]
 [-1.03320297e+00 -1.04144792e+00 -1.02559538e+00]
 [ 2.81233893e-02  5.37751472e-02  5.06977336e-05]
 [ 1.26530053e+00  1.25661142e+00  1.28007592e+00]
 [ 5.56764974e-01  5.66143405e-01  5.25548623e-01]]
```

[53]:
```
# Creating the contingency table
contingency_table = pd.crosstab(data['LandAndOceanAverageTemperature'],␣
 ↪data['LandAndOceanAverageTemperatureUncertainty'])

# Displaying the contingency table
print("Contingency Table:")
print(contingency_table)
```

```
Contingency Table:
LandAndOceanAverageTemperatureUncertainty  0.042  0.043  0.045  0.046  0.047  \
LandAndOceanAverageTemperature
12.475                                         0      0      0      0      0
12.620                                         0      0      0      0      0
12.658                                         0      0      0      0      0
12.702                                         0      0      0      0      0
12.732                                         0      0      0      0      0
…                                            …      …      …      …      …
17.578                                         0      0      0      0      0
17.589                                         0      0      0      0      0
17.607                                         0      0      0      0      0
17.609                                         0      0      0      0      0
17.611                                         0      0      0      0      0

LandAndOceanAverageTemperatureUncertainty  0.048  0.049  0.050  0.051  0.052  \
LandAndOceanAverageTemperature
12.475                                         0      0      0      0      0
12.620                                         0      0      0      0      0
12.658                                         0      0      0      0      0
12.702                                         0      0      0      0      0
12.732                                         0      0      0      0      0
```

```
…                                                …    …    …    …    …
17.578                                           0    0    0    0    0
17.589                                           0    0    0    0    0
17.607                                           0    0    0    0    0
17.609                                           0    0    0    0    0
17.611                                           0    0    0    0    0

LandAndOceanAverageTemperatureUncertainty  …  0.378  0.387  0.389  0.402  \
LandAndOceanAverageTemperature             …
12.475                                     …      0      0      0      0
12.620                                     …      0      0      0      0
12.658                                     …      1      0      0      0
12.702                                     …      0      0      0      0
12.732                                     …      0      0      0      0
…                                          …    …    …    …    …
17.578                                     …      0      0      0      0
17.589                                     …      0      0      0      0
17.607                                     …      0      0      0      0
17.609                                     …      0      0      0      0
17.611                                     …      0      0      0      0

LandAndOceanAverageTemperatureUncertainty  0.414  0.417  0.427  0.438  0.442  \
LandAndOceanAverageTemperature
12.475                                           0      0      0      0      0
12.620                                           0      0      0      0      0
12.658                                           0      0      0      0      0
12.702                                           0      0      0      0      0
12.732                                           0      0      0      0      0
…                                                …    …    …    …    …
17.578                                           0      0      0      0      0
17.589                                           0      0      0      0      0
17.607                                           0      0      0      0      0
17.609                                           0      0      0      0      0
17.611                                           0      0      0      0      0

LandAndOceanAverageTemperatureUncertainty  0.457
LandAndOceanAverageTemperature
12.475                                           0
12.620                                           0
12.658                                           0
12.702                                           0
12.732                                           0
…                                                …
17.578                                           0
17.589                                           0
17.607                                           0
17.609                                           0
17.611                                           0
```

```
[1596 rows x 294 columns]
```

[54]:
```
# Counting occurrences of combinations
occurrences = data.groupby(['LandAndOceanAverageTemperature',␣
 ↪'LandAndOceanAverageTemperatureUncertainty']).size()

# Displaying the occurrences
print("Occurrences of Combinations:")
print(occurrences)
```

```
Occurrences of Combinations:
LandAndOceanAverageTemperature  LandAndOceanAverageTemperatureUncertainty
12.475                          0.299                                       1
12.620                          0.187                                       1
12.658                          0.378                                       1
12.702                          0.186                                       1
12.732                          0.372                                       1
                                                                           ..
17.578                          0.067                                       1
17.589                          0.057                                       1
17.607                          0.064                                       1
17.609                          0.062                                       1
17.611                          0.058                                       1
Length: 1987, dtype: int64
```

[55]:
```
# Calculate conditional probabilities
conditional_probs = occurrences.div(occurrences.sum(level=0), axis=0)

# Display conditional probabilities
print("Conditional Probabilities:")
print(conditional_probs)
```

```
Conditional Probabilities:
LandAndOceanAverageTemperature  LandAndOceanAverageTemperatureUncertainty
12.475                          0.299                                     1.0
12.620                          0.187                                     1.0
12.658                          0.378                                     1.0
12.702                          0.186                                     1.0
12.732                          0.372                                     1.0
                                                                          ...
17.578                          0.067                                     1.0
17.589                          0.057                                     1.0
17.607                          0.064                                     1.0
17.609                          0.062                                     1.0
17.611                          0.058                                     1.0
Length: 1987, dtype: float64
```

```
<ipython-input-55-c0e50f176fed>:2: FutureWarning: Using the level keyword in
DataFrame and Series aggregations is deprecated and will be removed in a future
version. Use groupby instead. df.sum(level=1) should use
df.groupby(level=1).sum().
  conditional_probs = occurrences.div(occurrences.sum(level=0), axis=0)
```

[57]:
```
# Counting occurrences of combinations
occurrences = data.groupby(['LandAndOceanAverageTemperature',␣
 ↪'LandAndOceanAverageTemperatureUncertainty']).size()

# Calculate conditional probabilities
conditional_probs = occurrences.div(occurrences.groupby(level=0).sum())

# Display conditional probabilities
print("Conditional Probabilities:")
print(conditional_probs)
```

```
Conditional Probabilities:
LandAndOceanAverageTemperature  LandAndOceanAverageTemperatureUncertainty
12.475                          0.299                                        1.0
12.620                          0.187                                        1.0
12.658                          0.378                                        1.0
12.702                          0.186                                        1.0
12.732                          0.372                                        1.0
                                                                             ...
17.578                          0.067                                        1.0
17.589                          0.057                                        1.0
17.607                          0.064                                        1.0
17.609                          0.062                                        1.0
17.611                          0.058                                        1.0
Length: 1987, dtype: float64
```

[61]:
```
import pandas as pd
import numpy as np


# Reshape 'occurrences' into a DataFrame
occurrences_df = occurrences.unstack(fill_value=0)

# Compute row and column totals
row_totals = occurrences_df.sum(axis=1)
col_totals = occurrences_df.sum(axis=0)
total = occurrences_df.sum().sum()

# Calculate expected frequencies
expected = np.outer(row_totals, col_totals) / total
```

```
# Calculate adjusted residuals (Quetelet index)
residuals = (occurrences_df - expected) / np.sqrt(expected)

# Display adjusted residuals
print("Adjusted Residuals (Quetelet Index):")
print(residuals)
```

```
Adjusted Residuals (Quetelet Index):
LandAndOceanAverageTemperatureUncertainty     0.042     0.043     0.045  \
LandAndOceanAverageTemperature
12.475                                    -0.022406 -0.022406 -0.038808
12.620                                    -0.022406 -0.022406 -0.038808
12.658                                    -0.022406 -0.022406 -0.038808
12.702                                    -0.022406 -0.022406 -0.038808
12.732                                    -0.022406 -0.022406 -0.038808
…                                               …         …         …
17.578                                    -0.022406 -0.022406 -0.038808
17.589                                    -0.022406 -0.022406 -0.038808
17.607                                    -0.022406 -0.022406 -0.038808
17.609                                    -0.022406 -0.022406 -0.038808
17.611                                    -0.022406 -0.022406 -0.038808

LandAndOceanAverageTemperatureUncertainty     0.046     0.047     0.048  \
LandAndOceanAverageTemperature
12.475                                    -0.038808 -0.044811 -0.077615
12.620                                    -0.038808 -0.044811 -0.077615
12.658                                    -0.038808 -0.044811 -0.077615
12.702                                    -0.038808 -0.044811 -0.077615
12.732                                    -0.038808 -0.044811 -0.077615
…                                               …         …         …
17.578                                    -0.038808 -0.044811 -0.077615
17.589                                    -0.038808 -0.044811 -0.077615
17.607                                    -0.038808 -0.044811 -0.077615
17.609                                    -0.038808 -0.044811 -0.077615
17.611                                    -0.038808 -0.044811 -0.077615

LandAndOceanAverageTemperatureUncertainty     0.049     0.050     0.051  \
LandAndOceanAverageTemperature
12.475                                    -0.080784 -0.112028 -0.105091
12.620                                    -0.080784 -0.112028 -0.105091
12.658                                    -0.080784 -0.112028 -0.105091
12.702                                    -0.080784 -0.112028 -0.105091
12.732                                    -0.080784 -0.112028 -0.105091
…                                               …         …         …
17.578                                    -0.080784 -0.112028 -0.105091
17.589                                    -0.080784 -0.112028 -0.105091
17.607                                    -0.080784 -0.112028 -0.105091
```

```
17.609                                    -0.080784 -0.112028 -0.105091
17.611                                    -0.080784 -0.112028 -0.105091

LandAndOceanAverageTemperatureUncertainty     0.052  …      0.378     0.387  \
LandAndOceanAverageTemperature                       …
12.475                                    -0.116423  …  -0.031686 -0.022406
12.620                                    -0.116423  …  -0.031686 -0.022406
12.658                                    -0.116423  …  31.527781 -0.022406
12.702                                    -0.116423  …  -0.031686 -0.022406
12.732                                    -0.116423  …  -0.031686 -0.022406
…                                             …  …        …        …
17.578                                    -0.116423  …  -0.031686 -0.022406
17.589                                    -0.116423  …  -0.031686 -0.022406
17.607                                    -0.116423  …  -0.031686 -0.022406
17.609                                    -0.116423  …  -0.031686 -0.022406
17.611                                    -0.116423  …  -0.031686 -0.022406

LandAndOceanAverageTemperatureUncertainty     0.389     0.402     0.414  \
LandAndOceanAverageTemperature
12.475                                    -0.022406 -0.022406 -0.022406
12.620                                    -0.022406 -0.022406 -0.022406
12.658                                    -0.022406 -0.022406 -0.022406
12.702                                    -0.022406 -0.022406 -0.022406
12.732                                    -0.022406 -0.022406 -0.022406
…                                             …       …       …
17.578                                    -0.022406 -0.022406 -0.022406
17.589                                    -0.022406 -0.022406 -0.022406
17.607                                    -0.022406 -0.022406 -0.022406
17.609                                    -0.022406 -0.022406 -0.022406
17.611                                    -0.022406 -0.022406 -0.022406

LandAndOceanAverageTemperatureUncertainty     0.417     0.427     0.438  \
LandAndOceanAverageTemperature
12.475                                    -0.022406 -0.022406 -0.022406
12.620                                    -0.022406 -0.022406 -0.022406
12.658                                    -0.022406 -0.022406 -0.022406
12.702                                    -0.022406 -0.022406 -0.022406
12.732                                    -0.022406 -0.022406 -0.022406
…                                             …       …       …
17.578                                    -0.022406 -0.022406 -0.022406
17.589                                    -0.022406 -0.022406 -0.022406
17.607                                    -0.022406 -0.022406 -0.022406
17.609                                    -0.022406 -0.022406 -0.022406
17.611                                    -0.022406 -0.022406 -0.022406

LandAndOceanAverageTemperatureUncertainty     0.442     0.457
LandAndOceanAverageTemperature
12.475                                    -0.022406 -0.022406
```

```
12.620                                    -0.022406 -0.022406
12.658                                    -0.022406 -0.022406
12.702                                    -0.022406 -0.022406
12.732                                    -0.022406 -0.022406
…                                               …         …
17.578                                    -0.022406 -0.022406
17.589                                    -0.022406 -0.022406
17.607                                    -0.022406 -0.022406
17.609                                    -0.022406 -0.022406
17.611                                    -0.022406 -0.022406

[1596 rows x 294 columns]
```

[64]:
```
abs_residuals = residuals.abs()
```

[67]:
```
average_abs_residuals = abs_residuals.values.mean()
print(f"average_abs_residuals: {average_abs_residuals}")
```

```
average_abs_residuals: 0.10802117764625002
```

[68]:
```
from scipy.stats import chi2_contingency


chi2_stat, p_val, _, _ = chi2_contingency(contingency_table)

print(f"Chi-squared Statistic: {chi2_stat}")
```

```
Chi-squared Statistic: 466644.89099518606
```

[69]:
```
# Calculate chi-squared statistic
chi2_stat, p_val, dof, _ = chi2_contingency(contingency_table)

# Calculate the number of observations
num_observations = contingency_table.size

# Compute the average chi-squared value per observation
avg_chi2_per_observation = chi2_stat / num_observations

# Print the result
print(f"Average chi-squared value per observation: {avg_chi2_per_observation}")
```

```
Average chi-squared value per observation: 0.9945034588920986
```

[73]:
```
from scipy.stats import chi2_contingency

# Assuming contingency_table is your previously generated contingency table
chi2_statistic, p_value, dof, expected = chi2_contingency(contingency_table)
```

```
print(f"Chi-squared statistic: {chi2_statistic}")
print(f"Degrees of freedom: {dof}")
```

```
Chi-squared statistic: 466644.89099518606
Degrees of freedom: 467335
```

[75]:
```
from scipy.stats import chi2

# Degrees of freedom
n_rows = contingency_table.shape[0]
n_cols = contingency_table.shape[1]
degrees_of_freedom = (n_rows - 1) * (n_cols - 1)

# Critical chi-squared values for 95% and 99% confidence levels
alpha_95 = 0.05
alpha_99 = 0.01

critical_value_95 = chi2.ppf(1 - alpha_95, degrees_of_freedom)
critical_value_99 = chi2.ppf(1 - alpha_99, degrees_of_freedom)

print(f"Degrees of freedom: {degrees_of_freedom}")
print(f"Critical value for 95% confidence level: {critical_value_95}")
print(f"Critical value for 99% confidence level: {critical_value_99}")

# Check statistical significance
if chi2_statistic > critical_value_95:
    print("Statistically significant association at 95% confidence level")
else:
    print("No statistically significant association at 95% confidence level")

if chi2_statistic > critical_value_99:
    print("Statistically significant association at 99% confidence level")
else:
    print("No statistically significant association at 99% confidence level")
```

```
Degrees of freedom: 467335
Critical value for 95% confidence level: 468926.35327760974
Critical value for 99% confidence level: 469587.01517297537
No statistically significant association at 95% confidence level
No statistically significant association at 99% confidence level
```

[77]:
```
import numpy as np

# Assuming you have the standardized feature values in 'selected_features'
selected_features = np.array([-2.85358528e-17, 3.56698161e-17, -7.13396321e-18])

# Number of bootstrap iterations
```

```
num_iterations = 1000  # You can adjust this as needed

# Store resampled grand means
resampled_means = []

for _ in range(num_iterations):
    # Generate a resampled dataset by sampling with replacement
    resampled_data = np.random.choice(selected_features,␣
 ↪size=len(selected_features), replace=True)

    # Compute the grand mean for this resampled dataset
    resampled_mean = np.mean(resampled_data)

    # Store the resampled grand mean
    resampled_means.append(resampled_mean)

# Calculate the confidence interval (95%)
confidence_interval = np.percentile(resampled_means, [2.5, 97.5])

print(f"95% Confidence Interval for Grand Mean: {confidence_interval}")
```

```
95% Confidence Interval for Grand Mean: [-2.85358528e-17  3.56698161e-17]
```

[79]:
```
import numpy as np

# Consider 'grand_means' as an array of grand mean values obtained from␣
 ↪bootstrap resampling
grand_means = np.array([-2.85358528e-17, 3.56698161e-17, -7.13396321e-18])

# Calculate the pivotal confidence interval (95%)
confidence_interval_pivotal = np.percentile(grand_means, [2.5, 97.5])

print(f"Pivotal 95% Confidence Interval for Grand Mean:␣
 ↪{confidence_interval_pivotal}")
```

```
Pivotal 95% Confidence Interval for Grand Mean: [-2.74657583e-17
3.35296271e-17]
```

[80]:
```
import numpy as np

# Consider 'grand_means' as an array of grand mean values obtained from␣
 ↪bootstrap resampling
grand_means = np.array([-2.85358528e-17, 3.56698161e-17, -7.13396321e-18])

# Calculate mean and standard deviation of the grand mean
mean_grand_mean = np.mean(grand_means)
```

```
std_grand_mean = np.std(grand_means, ddof=1)  # Use ddof=1 for sample standard␣
 ↪deviation

# Compute the non-pivotal confidence interval assuming normal distribution (95%)
z_score = 1.96  # Corresponds to the 95% confidence level
confidence_interval_non_pivotal = [mean_grand_mean - z_score * std_grand_mean,
                                   mean_grand_mean + z_score * std_grand_mean]

print(f"Non-pivotal 95% Confidence Interval for Grand Mean:␣
 ↪{confidence_interval_non_pivotal}")
```

```
Non-pivotal 95% Confidence Interval for Grand Mean: [-6.407617576236292e-17,
6.407617582236291e-17]
```

[83]:
```
num_clusters = 5  # Set the number of clusters you want

# Rest of your code here
kmeans = KMeans(n_clusters=num_clusters)
# ... (rest of your code)
```

[86]:
```
from sklearn.cluster import KMeans
import numpy as np

# Assuming 'cleaned_data' contains your dataset and 'feature_of_interest' is a␣
 ↪column you want to analyze
num_clusters = 5  # Define the number of clusters you want

# Initializing and fitting KMeans
kmeans = KMeans(n_clusters=num_clusters)
kmeans.fit(cleaned_data)

# Get the cluster labels and add them to your dataset as a new column
cleaned_data['cluster_label'] = kmeans.labels_

# Extract feature data for each cluster
cluster_data = []
for cluster_id in range(num_clusters):
    feature_data_cluster = cleaned_data[cleaned_data['cluster_label'] ==␣
 ↪cluster_id]['LandMinTemperature']
    cluster_data.append(feature_data_cluster)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```

[90]:
```
# Check if the confidence intervals obtained from both methods suggest a␣
 ↪significant difference
if confidence_interval_pivotal[0] > 0 or confidence_interval_pivotal[1] < 0:
    print("Pivotal method suggests a statistically significant difference.")
else:
    print("Pivotal method does not show a statistically significant difference.
 ↪")

if confidence_interval_non_pivotal[0] > 0 or confidence_interval_non_pivotal[1]␣
 ↪< 0:
    print("Non-pivotal method suggests a statistically significant difference.")
else:
    print("Non-pivotal method does not show a statistically significant␣
 ↪difference.")
```

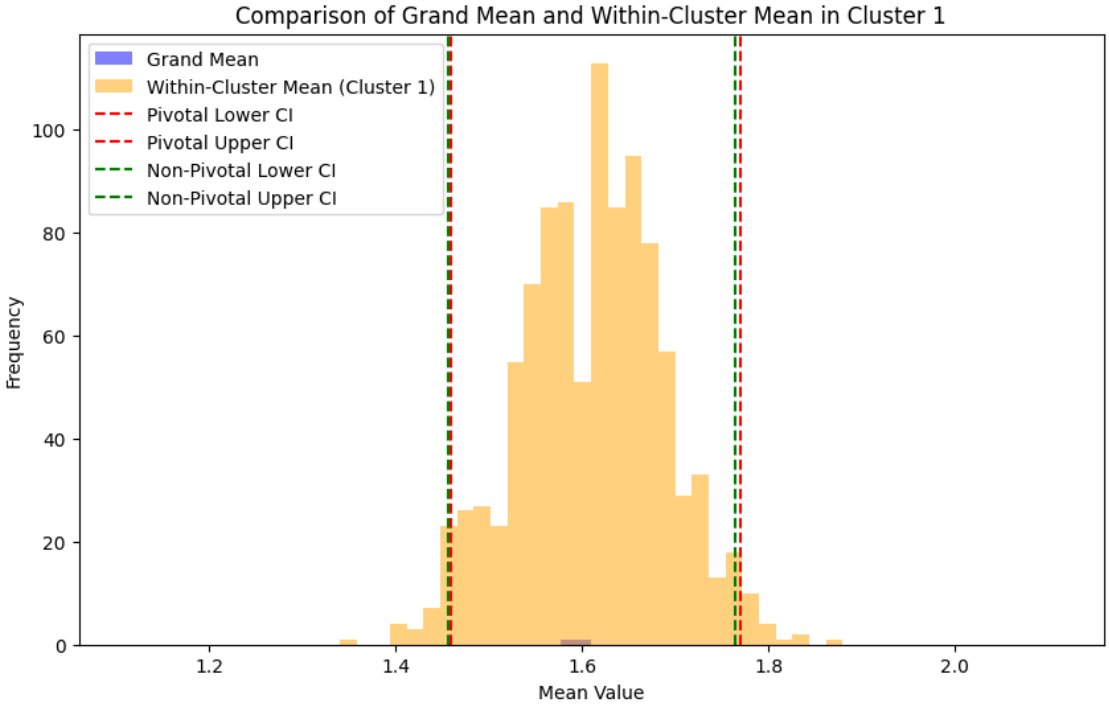
```
Pivotal method does not show a statistically significant difference.
Non-pivotal method does not show a statistically significant difference.
```

[92]:
```
import numpy as np

# Suppose 'cluster1_data' contains the feature data for Cluster 1
bootstrap_samples = 1000  # Number of bootstrap samples
cluster1_data = [1.2, 1.5, 1.7, 1.3, 1.9, 2.0, 1.6, 1.8, 1.4, 1.7]
# Original data
cluster1_original_data = np.array(cluster1_data)

# Grand mean of the entire dataset (including both clusters)
grand_mean = np.mean(cluster1_original_data)

# Bootstrap resampling for the within-cluster mean of Cluster 1
within_means_cluster1 = []

for _ in range(bootstrap_samples):
    # Resample with replacement for Cluster 1 and calculate the mean
    resampled_data_cluster1 = np.random.choice(cluster1_data,␣
 ↪size=len(cluster1_data), replace=True)
    mean_cluster1 = np.mean(resampled_data_cluster1)
    within_means_cluster1.append(mean_cluster1)

# Pivotal approach: calculate confidence interval for within-cluster mean
confidence_interval_within_pivotal = np.percentile(within_means_cluster1, [2.5,␣
 ↪97.5])

# Non-pivotal approach: estimate confidence interval using standard deviation
mean_within = np.mean(within_means_cluster1)
std_within = np.std(within_means_cluster1, ddof=1)  # Use ddof=1 for sample␣
 ↪standard deviation
```

```
z_score = 1.96  # Corresponds to the 95% confidence level
confidence_interval_within_non_pivotal = [mean_within - z_score * std_within,␣
 ↪mean_within + z_score * std_within]

print(f"Grand Mean of Cluster 1: {grand_mean}")
print(f"Pivotal 95% Confidence Interval for Within-Cluster Mean:␣
 ↪{confidence_interval_within_pivotal}")
print(f"Non-pivotal 95% Confidence Interval for Within-Cluster Mean:␣
 ↪{confidence_interval_within_non_pivotal}")
```

```
Grand Mean of Cluster 1: 1.61
Pivotal 95% Confidence Interval for Within-Cluster Mean: [1.46 1.77]
Non-pivotal 95% Confidence Interval for Within-Cluster Mean:
[1.4570983904089831, 1.7644216095910168]
```

[93]:
```
import matplotlib.pyplot as plt



# Create histograms for the grand mean and within-cluster mean in Cluster 1
plt.figure(figsize=(10, 6))

# Histogram for the grand mean
plt.hist([grand_mean], bins=30, alpha=0.5, color='blue', label='Grand Mean')

# Histogram for the within-cluster mean in Cluster 1
plt.hist(within_means_cluster1, bins=30, alpha=0.5, color='orange',␣
 ↪label='Within-Cluster Mean (Cluster 1)')

# Plot vertical lines for pivotal confidence intervals
plt.axvline(x=confidence_interval_within_pivotal[0], color='red',␣
 ↪linestyle='--', label='Pivotal Lower CI')
plt.axvline(x=confidence_interval_within_pivotal[1], color='red',␣
 ↪linestyle='--', label='Pivotal Upper CI')

# Plot vertical lines for non-pivotal confidence intervals
plt.axvline(x=confidence_interval_within_non_pivotal[0], color='green',␣
 ↪linestyle='--', label='Non-Pivotal Lower CI')
plt.axvline(x=confidence_interval_within_non_pivotal[1], color='green',␣
 ↪linestyle='--', label='Non-Pivotal Upper CI')

plt.xlabel('Mean Value')
plt.ylabel('Frequency')
plt.title('Comparison of Grand Mean and Within-Cluster Mean in Cluster 1')
plt.legend()
plt.show()
```


Comparison of Grand Mean and Within-Cluster Mean in Cluster 1
Grand Mean
Within-Cluster Mean (Cluster 1)
Pivotal Lower CI
Pivotal Upper CI
Non-Pivotal Lower CI
Non-Pivotal Upper CI
Frequency
0
20
40
60
80
100
1.2
1.4
1.6
1.8
2.0
Mean Value