Technical Report: Customer Satisfaction Factor Analysis

1. Executive Summary

This report details the technical implementation of a Factor Analysis performed on customer satisfaction data. The objective was to reduce the dimensionality of 23 survey variables into a smaller set of interpretable latent factors to guide business strategy. The analysis successfully identified 5 key factors driving customer satisfaction.

2. Technical Methodology

The analysis was implemented using Python, leveraging the `pandas`, `matplotlib`, `seaborn`, and `factor_analyzer` libraries. The process followed a standard statistical workflow:

2.1 Data Loading and Preprocessing

Data is loaded from a CSV file into a pandas DataFrame. Preprocessing involves handling missing values and ensuring data quality before analysis.

```
import pandas as pd

def load_data(filepath):
    df = pd.read_csv(filepath)
    return df
```

2.2 Correlation Analysis

Before performing factor analysis, we assessed the suitability of the data by examining the correlation matrix. High correlations between variables suggest the existence of underlying factors.

```
import seaborn as sns
import matplotlib.pyplot as plt

def generate_correlation_heatmap(df):
    plt.figure(figsize=(12, 10))
    corr = df.corr()
    sns.heatmap(corr, annot=False, cmap='RdBu_r', center=0)
    plt.title('Correlation Heatmap')
    plt.tight_layout()
    plt.show()
    plt.close()
```

2.3 Factor Selection (Scree Plot)

To determine the optimal number of factors, we used the Kaiser Criterion (eigenvalues > 1) and visualized the eigenvalues using a Scree Plot.

```
 from factor_analyzer import FactorAnalyzer
from sklearn.preprocessing import StandardScaler

def generate_scree_plot(df):
    # Standardization is crucial for Factor Analysis
    scaler = StandardScaler()
    df_scaled = scaler.fit_transform(df)

    fa = FactorAnalyzer(n_factors=10, rotation=None)
    fa.fit(df_scaled)
    ev, v = fa.get_eigenvalues()

    plt.figure(figsize=(8, 5))
    plt.plot(range(1, len(ev)+1), ev, marker='o')
    plt.title('Scree Plot (Kaiser Criterion)')
    plt.xlabel('Factor')
    plt.ylabel('Eigenvalue')
    plt.axhline(y=1, color='r', linestyle='--')
    plt.grid(True)
    plt.show()
    plt.close()
    return ev
```

2.4 Factor Extraction and Rotation

We employed Principal Axis Factoring with Varimax rotation. Varimax rotation was chosen to maximize the variance of squared loadings, resulting in a "simple structure" where each variable loads highly on only one factor, improving interpretability.

```
def generate_factor_loadings_plot(df, n_factors):
    scaler = StandardScaler()
    df_scaled = scaler.fit_transform(df)

    # Varimax rotation for orthogonal factors
    fa = FactorAnalyzer(n_factors=n_factors, rotation='varimax')
    fa.fit(df_scaled)
    loadings = pd.DataFrame(fa.loadings_, index=df.columns)

    plt.figure(figsize=(10, 12))
    sns.heatmap(loadings, annot=True, cmap='RdBu_r', center=0)
    plt.title(f'Factor Loadings ({n_factors} Factors)')
    plt.tight_layout()
    plt.show()
    plt.close()
```

2.5 Result Presentation

To make the results actionable, we implemented a formatting function to highlight significant factor loadings (absolute value > 0.4) and suppress noise.

```
import numpy as np

def format_factor_loadings(df, threshold=0.4):
    """
    Formats the factor loadings dataframe with colors.
    Values below the threshold are hidden.
    Significant values are highlighted in green.
    """
    df_presentation = df.copy()
    # Mask values below threshold
    df_presentation[np.abs(df_presentation) < threshold] = np.nan

    def highlight_high(val):
        if pd.isna(val):
            return ''
        return 'background-color: #8fbc8f' # DarkSeaGreen

    # Apply styling
    styler = df_presentation.style
    if hasattr(styler, 'map'):
        styler = styler.map(highlight_high)
    else:
        styler = styler.applymap(highlight_high)

    return styler.format(precision=3, na_rep="")
```

3. Analysis Results & Interpretation

The technical analysis yielded a robust 5-factor solution:

1. **Technical Excellence**: High loadings on technical expertise and problem-solving variables.
2. **Value & Pricing**: Driven by cost transparency and ROI demonstration.
3. **Relationship & Trust**: Centered on account management and long-term partnership.
4. **Project Management**: Defined by adherence to timelines and budget control.
5. **Support & Service**: Characterized by responsiveness and training quality.

These factors explain the majority of the variance in the dataset and provide clear, orthogonal dimensions for business optimization.

4. Conclusion

The implemented Python workflow provides a reproducible and scalable method for analyzing customer satisfaction. The use of `FactorAnalyzer` combined with custom visualization and formatting functions ensures that the complex statistical output is translated into clear, actionable business insights.