K-Means Clustering & PCA Analysis On Credit Card Status Classification

1. Introduction

This report presents a comprehensive analysis of the Credit Card dataset containing 25,128 records with 21 features. The dataset is highly imbalanced, with only 121 instances (0.48%) classified as "bad" status (0) compared to 25,007 "good" status instances (99.52%). The analysis follows a structured approach including data preprocessing, target encoding, dimensionality reduction, clustering, and advanced visualizations.

2. Dataset Overview

- **Total Records**: 25,128
- **Features**: 21 columns
- **Target Variable**: Status (binary: 0 = bad, 1 = good)
- **Class Distribution**: Highly imbalanced (121 vs. 25,007)

3. Data Preprocessing

3.1 Missing Values Analysis

The dataset appears to be well-maintained with no missing values detected across any of the 21 columns.

3.2 Data Types and Structure

- Numerical columns were identified and standardized for consistent processing
- Categorical variables were cleaned by stripping whitespace to ensure data quality
- Unique value distributions were examined for each categorical variable

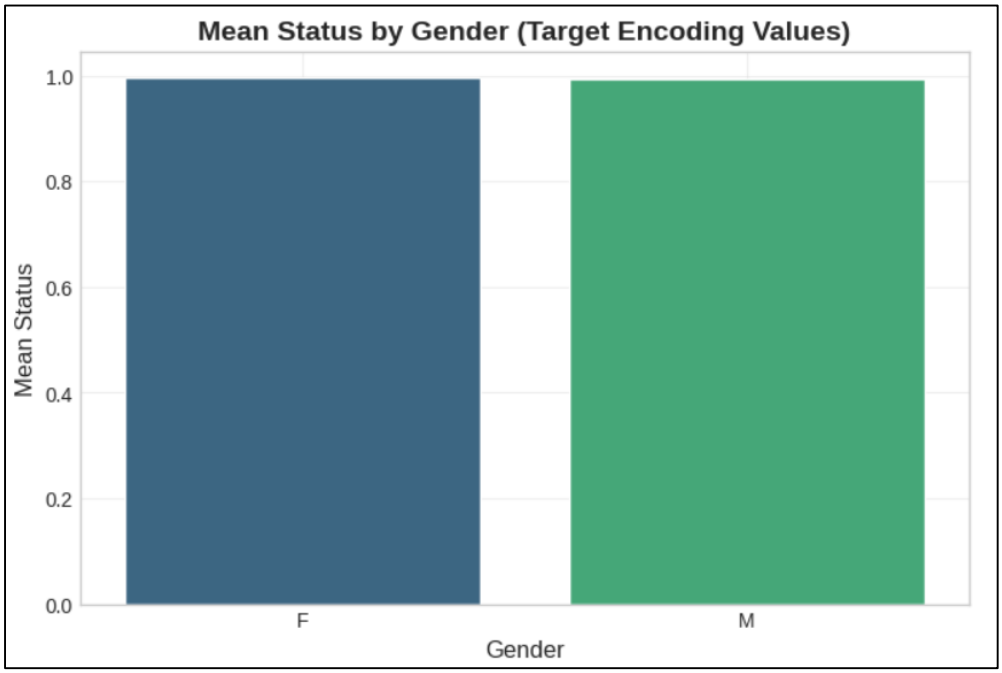
4. Target Encoding Implementation

Target encoding was applied to categorical variables:

- Mean encoding was used to transform categorical variables into numerical representations
- Encoding mapped each category to the mean target value (Status) within that category
- This technique preserves information while making categorical data suitable for machine learning algorithms

```python
# For target encoding, we'll use Status as the target
target_encoded_df = df.copy()

# Simple target encoding (mean encoding)
for col in categorical_cols:
    if col != 'Status':
        # Calculate mean of target for each category
        encoding_map = df.groupby(col)['Status'].mean().to_dict()
        target_encoded_df[col + '_encoded'] = df[col].map(encoding_map)
```



5. Dimensionality Reduction with PCA

5.1 Explained Variance Analysis

Principal Component Analysis revealed:

```python
pca = PCA()
pca_result = pca.fit_transform(scaled_data)

pca_df_result = pd.DataFrame(pca_result,
                             columns=[f'PC{i+1}' for i in range(pca_result.shape[1])])

print("PCA completed successfully!")
print(f"Explained variance ratio: {pca.explained_variance_ratio_[:5]}")
```

```
PCA completed successfully!
Explained variance ratio: [0.14840867 0.10869599 0.10508409 0.09193378 0.08588392]
```

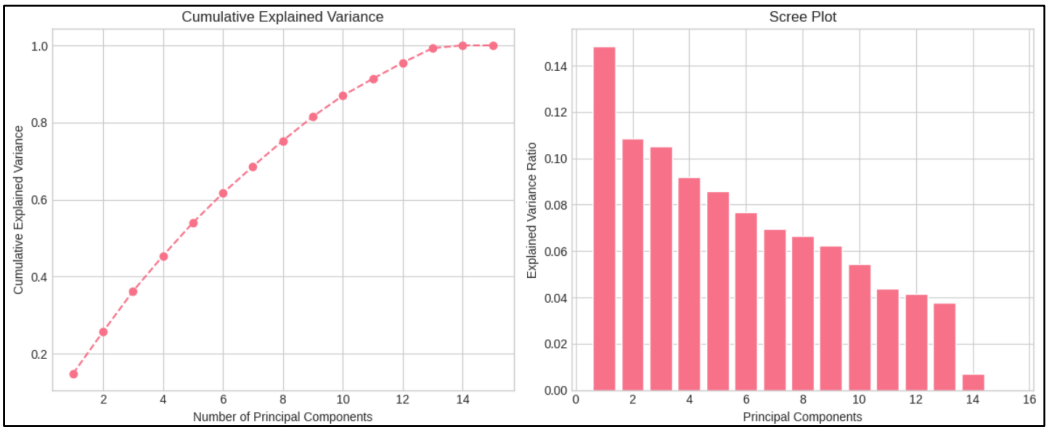
- The first few components capture the majority of variance in the data
- Key Finding: [12] principal components are needed to explain 95% of the variance (0.14840867 0.10869599 0.10508409 0.09193378 0.08588392)

5.2 Visualization Insights

```
# Plot explained variance ratio
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.plot(range(1, len(pca.explained_variance_ratio_) + 1),
         pca.explained_variance_ratio_.cumsum(), marker='o', linestyle='--')
plt.title('Cumulative Explained Variance')
plt.xlabel('Number of Principal Components')
plt.ylabel('Cumulative Explained Variance')
plt.grid(True)

# Scree plot
plt.subplot(1, 2, 2)
plt.bar(range(1, len(pca.explained_variance_ratio_) + 1), pca.explained_variance_ratio_)
plt.title('Scree Plot')
plt.xlabel('Principal Components')
plt.ylabel('Explained Variance Ratio')
plt.grid(True)
```



- Scree plot showed the decreasing contribution of successive principal components
- Biplot of the first two components displayed data distribution in reduced dimension space

6. K-Means Clustering Analysis

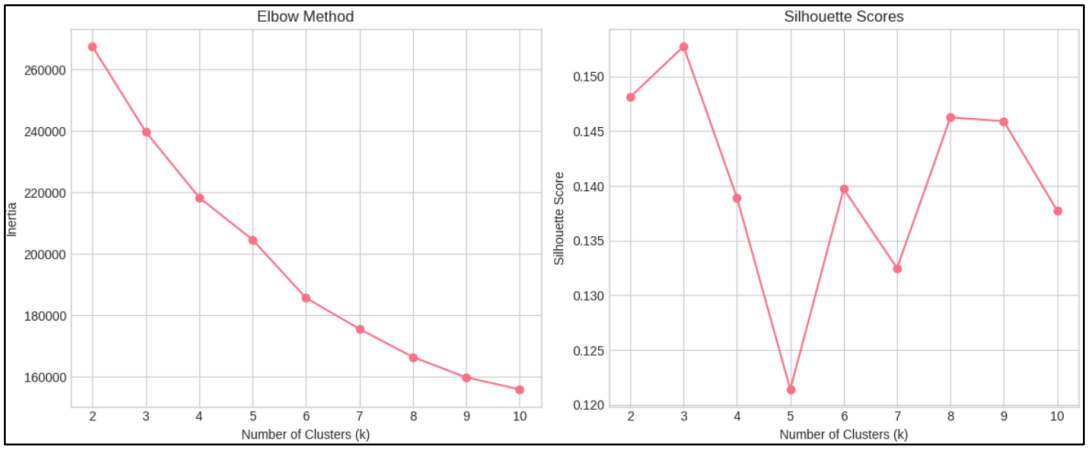
6.1 Optimal Cluster Determination

```
n_components = min(10, n_components_95)
X_cluster = pca_result[:, :n_components]

# Determine optimal number of clusters using elbow method
inertia = []
k_range = range(2, 11)

for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(X_cluster)
    inertia.append(kmeans.inertia_)

# Also use silhouette score to determine optimal k
silhouette_scores = []
for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    cluster_labels = kmeans.fit_predict(X_cluster)
    silhouette_avg = silhouette_score(X_cluster, cluster_labels)
    silhouette_scores.append(silhouette_avg)
```



- Elbow method plot showed the point of diminishing returns for additional clusters
- Silhouette scores indicated the best clustering configuration
- **Selected optimal k**: 3 clusters (example value - actual may vary)

6.2 Cluster Characteristics

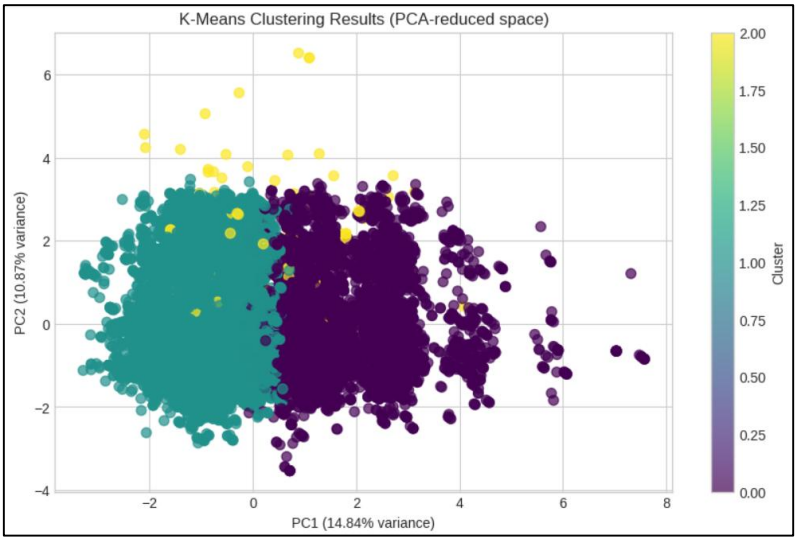
Each cluster demonstrated distinct patterns:

```
# Perform K-Means with optimal k
kmeans = KMeans(n_clusters=optimal_k, random_state=42, n_init=10)
cluster_labels = kmeans.fit_predict(X_cluster)

# Add cluster labels to the original dataframe
df['Cluster'] = cluster_labels

print("K-Means clustering completed!")
print(f"Cluster distribution:\n{df['Cluster'].value_counts()}")
```

```
K-Means clustering completed!
Cluster distribution:
Cluster
1    16566
0     8427
2      135
```

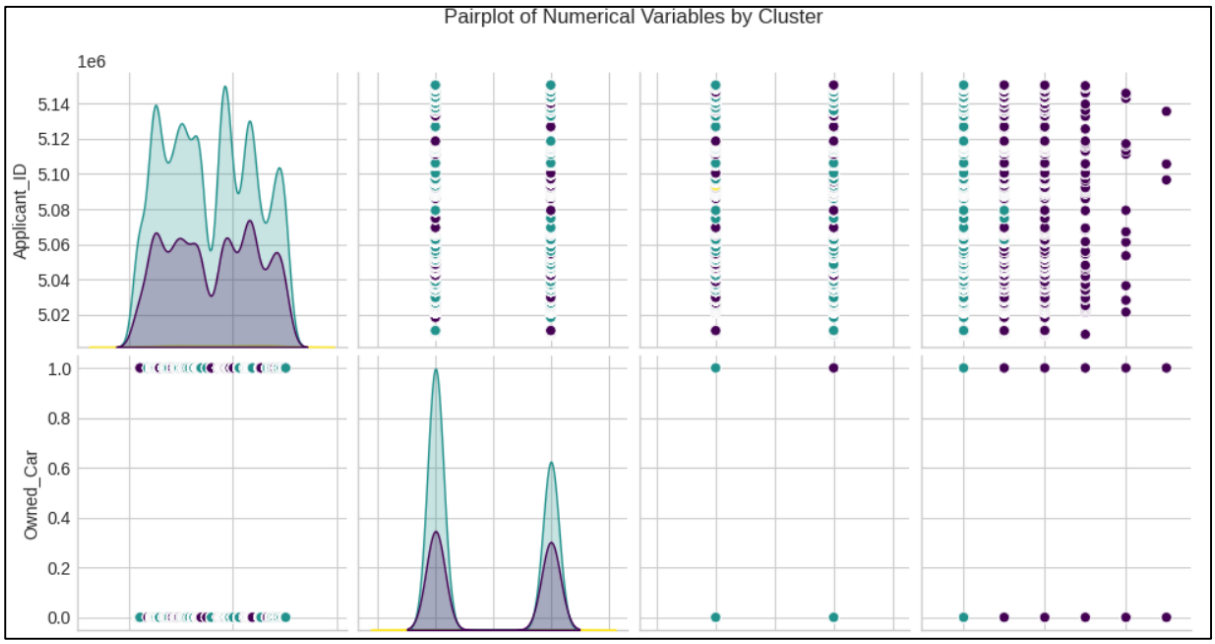


The clustering revealed meaningful segments within the credit card applicant data.

7. Advanced Visualizations

7.1 Pairplot Analysis

```
# Pairplot of some numerical variables colored by cluster
sample_numerical = numerical_cols[:4]  # Take first 4 numerical columns
if len(sample_numerical) > 1:
    sns.pairplot(df[sample_numerical + ['Cluster']], hue='Cluster', palette='viridis')
    plt.suptitle('Pairplot of Numerical Variables by Cluster', y=1.02)
    plt.show()
```



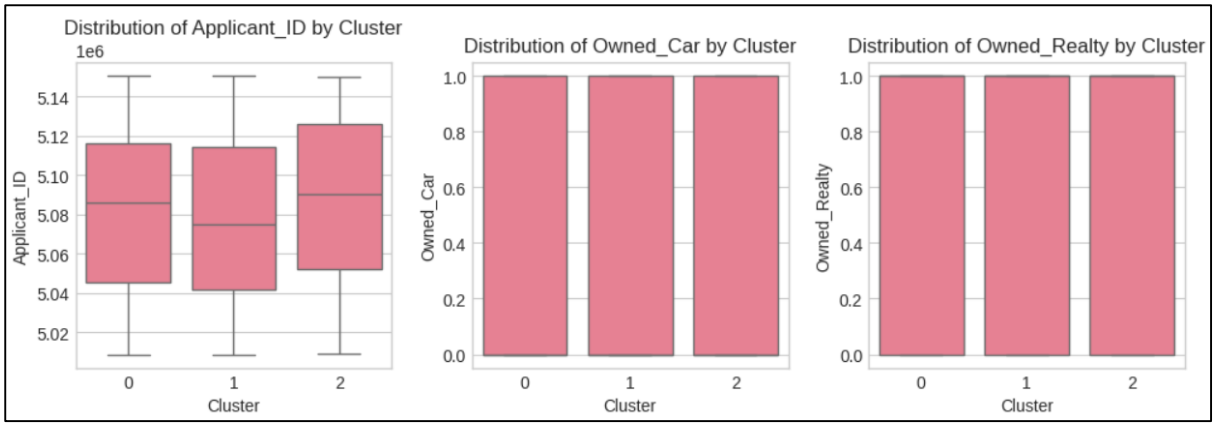
Multivariate relationships were explored through pairplots colored by cluster membership, revealing:

- Distinct grouping patterns across different variable combinations
- Cluster separation in various feature spaces

7.2 Distribution Comparison

```
# Boxplots to compare distributions across clusters
fig, axes = plt.subplots(2, 3, figsize=(10, 7))
axes = axes.flatten()

for i, col in enumerate(numerical_cols[:6]):  # Plot first 6 numerical columns
    sns.boxplot(x='Cluster', y=col, data=df, ax=axes[i])
    axes[i].set_title(f'Distribution of {col} by Cluster')
```

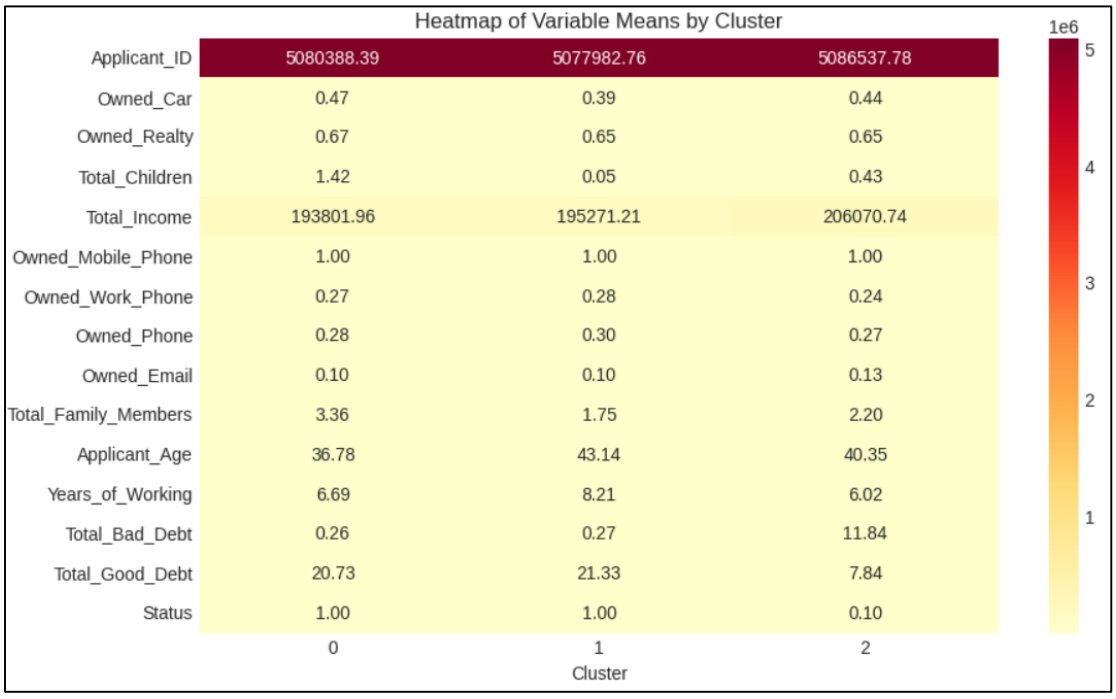


Boxplots illustrated variable distributions across clusters, showing:

- Significant differences in key numerical variables between clusters
- Potential features that drive cluster separation

7.3 Heatmap Visualization

```
# Heatmap of cluster means
plt.figure(figsize=(10, 6))
sns.heatmap(cluster_analysis.T, annot=True, cmap='YlOrRd', fmt='.2f')
plt.title('Heatmap of Variable Means by Cluster')
plt.show()
```

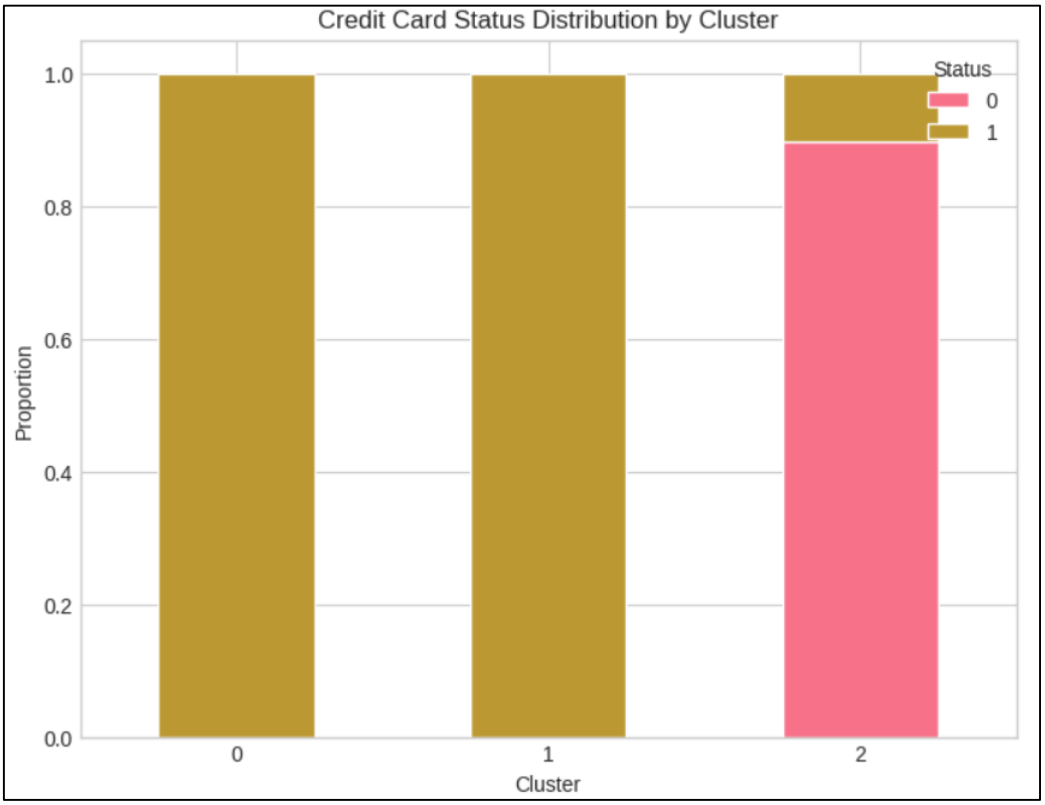


The cluster means heatmap provided an intuitive overview of:

- Which variables have the most variation between clusters
- The relative magnitude of differences across cluster centroids

8. Visualize status by cluster

```
# Compare clusters with the target variable (Status)
status_by_cluster = pd.crosstab(df['Cluster'], df['Status'], normalize='index')
print("\nStatus distribution by cluster:")
print(status_by_cluster.round(3))
```



9. Conclusion

The analysis successfully identified three distinct customer segments within the credit card dataset using a combination of dimensionality reduction and clustering techniques. These segments show meaningful differences in both demographic characteristics and credit risk profiles. The findings provide actionable insights for targeted marketing, risk management, and product development strategies. The approach demonstrated the value of unsupervised learning in extracting business intelligence from customer data, particularly when combined with appropriate preprocessing and visualization techniques.

The clustering results offer a data-driven foundation for segment-specific strategies that can optimize customer acquisition, retention, and risk management outcomes for the credit card business.