```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.impute import SimpleImputer
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, f1_s


# Load the dataset
df = pd.read_csv("labeled_dataset.csv")

# Define features and target columns
features = ["age", "email_breach_history", "transaction_amount", "distance", "interval",
categorical_features = ["transaction_type", "event_type", "email_domain", "phone_carrier"
target = "transaction_isFraud"


# Display data types
print("Data Types:\n", df.dtypes)

# Display unique value counts for each column
print("\nUnique Values per Column:\n", df.nunique())
```

```
Data Types:
                          object
name                      object
age                        int64
email_id                  object
email_domain              object
email_breach_history       int64
email_isValid               bool
phone_number               int64
phone_carrier             object
phone_disposableSim         bool
cluster_id                 int64
section_id                object
transaction_type          object
transaction_amount       float64
transaction_isFraud        int64
event_id                  object
event_type                object
screen_x                 float64
screen_y                 float64
distance                 float64
interval                   int64
speed                    float64
dtype: object

Unique Values per Column:
```

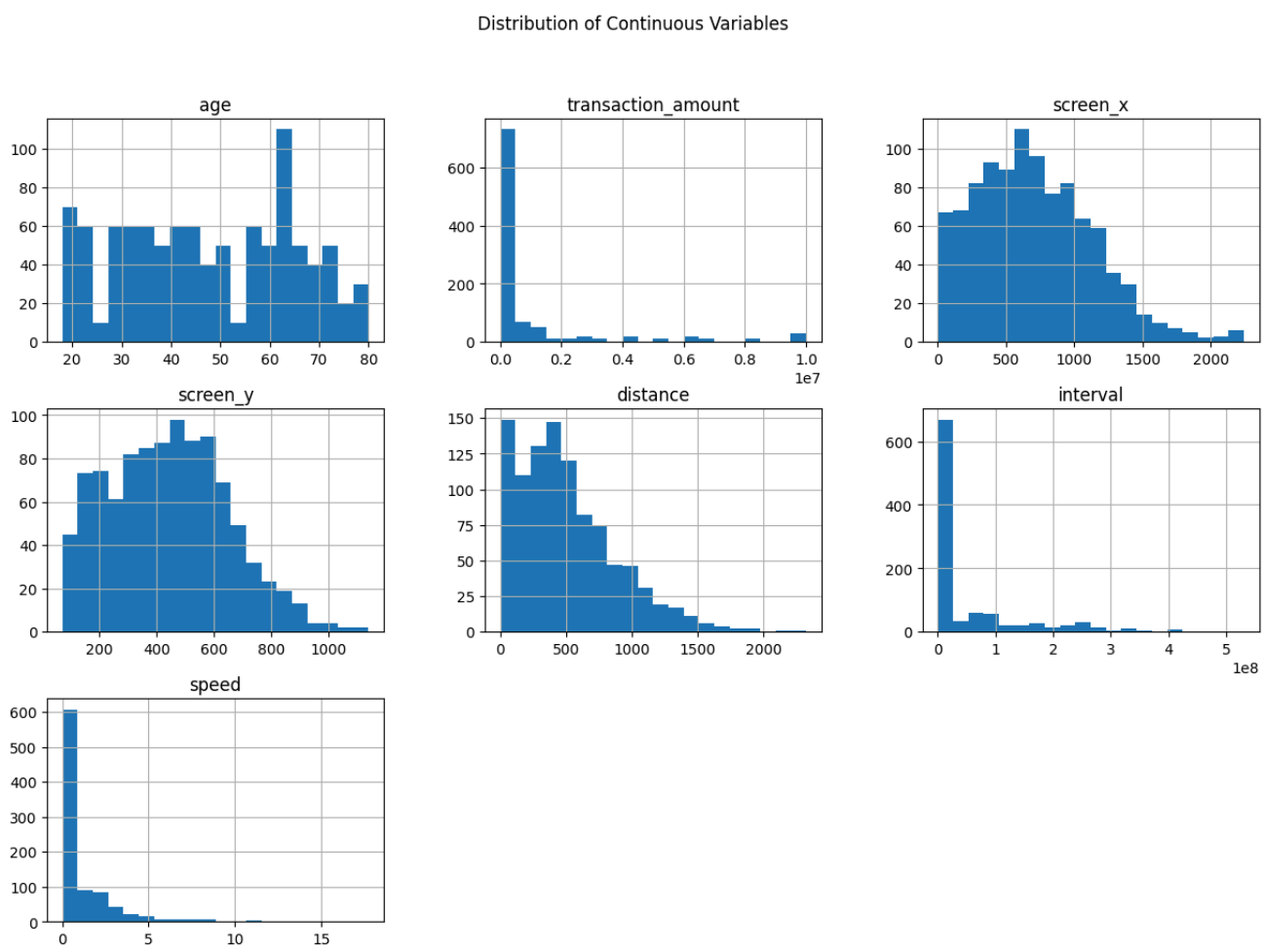
```
                            100
name                         67
age                          53
email_id                    100
email_domain                 10
email_breach_history          6
email_isValid                 2
phone_number                100
phone_carrier                 6
phone_disposableSim           2
cluster_id                    2
section_id                    2
transaction_type              4
transaction_amount           98
transaction_isFraud           2
event_id                      5
event_type                    4
screen_x                    801
screen_y                    730
distance                    899
interval                    579
speed                       834
dtype: int64
```

```
# Check for missing values
print("\nMissing Values:\n", df.isnull().sum())

# Summary statistics
print("\nSummary Statistics:\n", df.describe())
```

```
Missing Values:
                             0
name                         0
age                          0
email_id                     0
email_domain                 0
email_breach_history         0
email_isValid                0
phone_number                 0
phone_carrier                0
phone_disposableSim          0
cluster_id                   0
section_id                   0
transaction_type             0
transaction_amount           0
transaction_isFraud          0
event_id                     0
event_type                   0
screen_x                     0
screen_y                     0
distance                     0
interval                     0
speed                      100
dtype: int64

Summary Statistics:
                 age  email_domain  email_breach_history  phone_number  \
count  1000.000000   1000.000000           1000.000000  1.000000e+03
```

```
              mean     47.670000      4.300000              2.220000  5.561776e+09
              std      17.330939      2.939156              1.713048  2.499211e+09
              min      18.000000      0.000000              0.000000  1.122711e+09
              25%      33.000000      3.000000              1.000000  3.496550e+09
              50%      46.000000      4.000000              2.000000  5.986035e+09
              75%      63.000000      6.000000              4.000000  7.361472e+09
              max      80.000000      9.000000              5.000000  9.883821e+09

                     phone_carrier   cluster_id  transaction_type  transaction_amount  \
              count    1000.000000  1000.000000       1000.000000        1.000000e+03
              mean        2.310000     0.630000          1.570000        1.028864e+06
              std         1.765508     0.483046          0.982887        2.227298e+06
              min         0.000000     0.000000          0.000000        0.000000e+00
              25%         1.000000     0.000000          1.000000        3.170260e+04
              50%         2.000000     1.000000          1.000000        1.539827e+05
              75%         4.000000     1.000000          2.250000        6.221058e+05
              max         5.000000     1.000000          3.000000        1.000000e+07

                     transaction_isFraud   event_type     screen_x     screen_y  \
              count          1000.000000  1000.000000  1000.000000  1000.000000
              mean              0.450000     1.012000   706.539776   442.609938
              std               0.497743     0.878993   429.087679   207.679142
              min               0.000000     0.000000     0.000000    71.333336
              25%               0.000000     0.000000   378.916675   283.333340
              50%               0.000000     1.000000   659.600000   442.500000
              75%               1.000000     1.000000   982.000050   587.050065
              max               1.000000     3.000000  2239.980500  1141.000000

                       distance      interval       speed
```

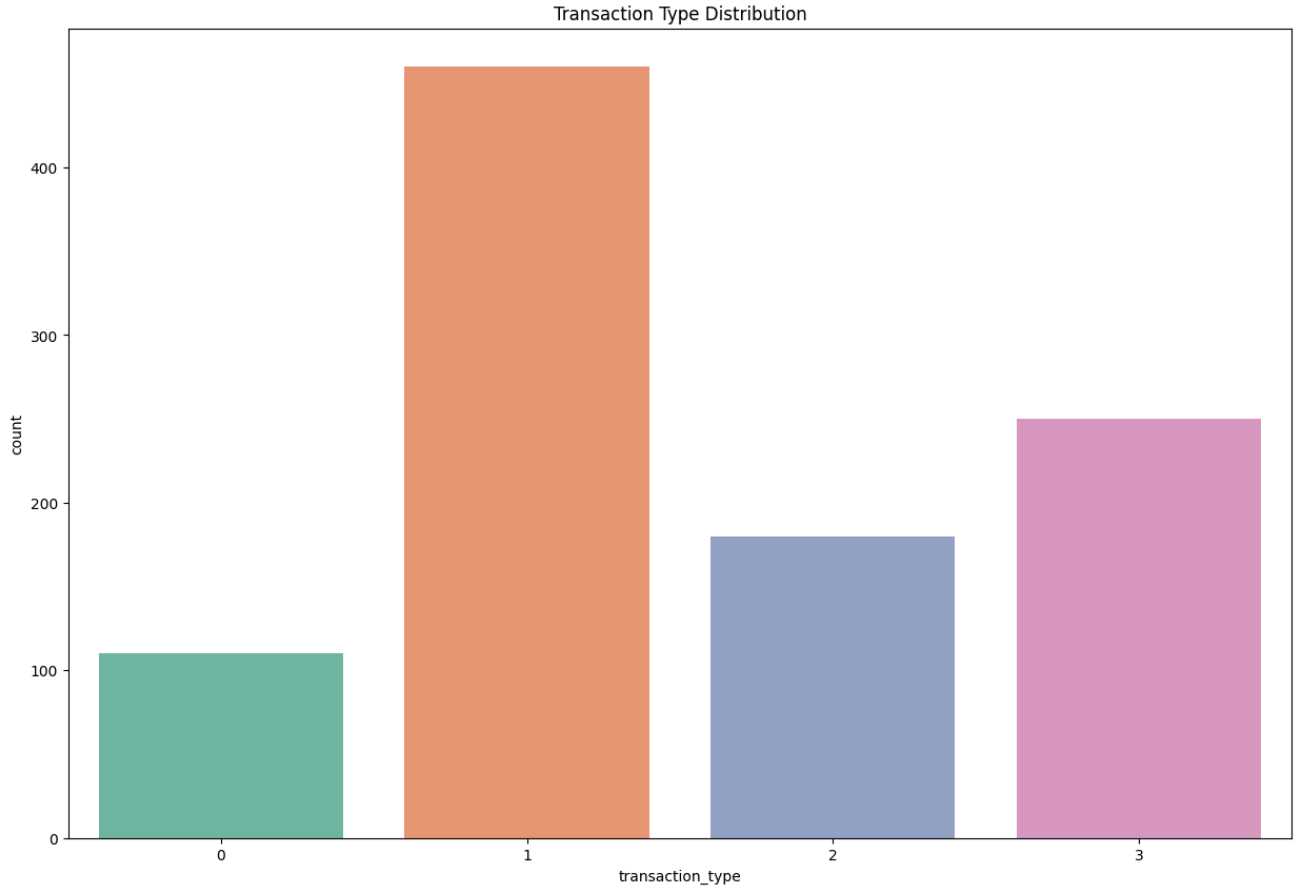
```
# Plot histograms for continuous variables
df[["age", "transaction_amount", "screen_x", "screen_y", "distance", "interval", "speed"]
plt.suptitle("Distribution of Continuous Variables")
plt.show()
```



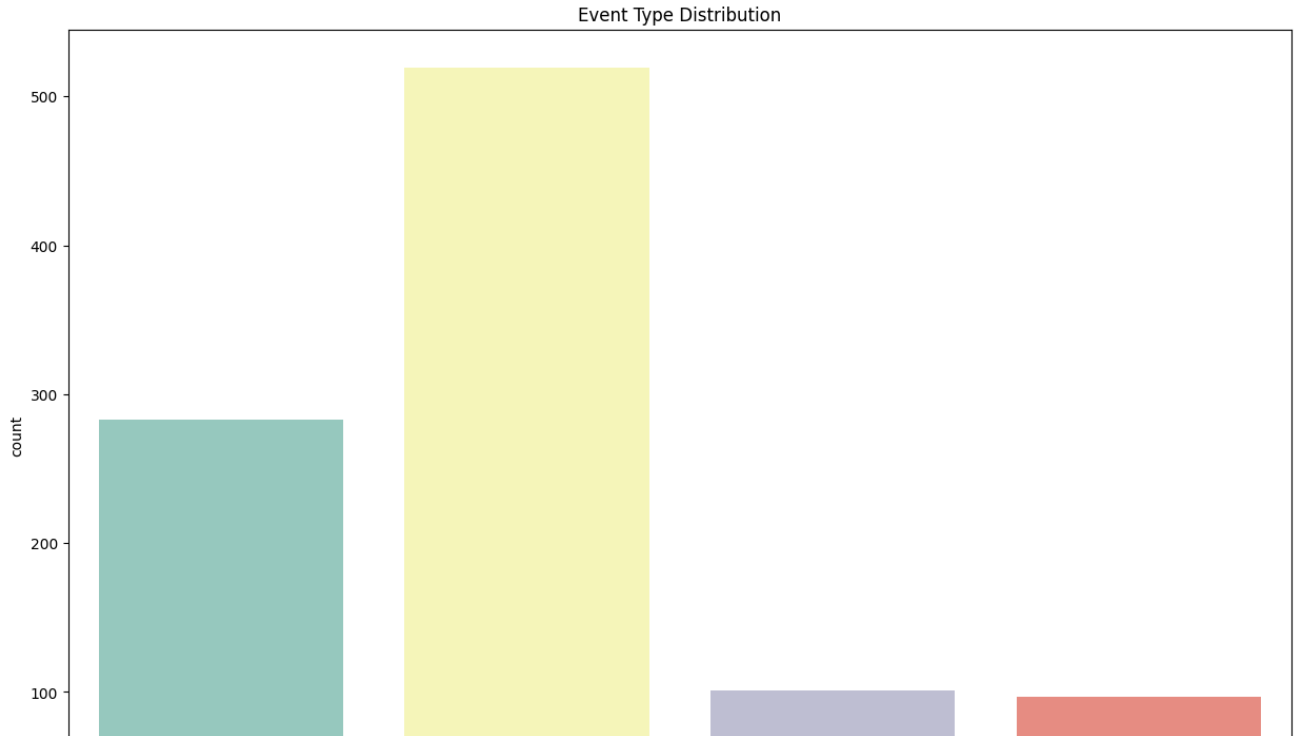

```
# Categorical variable count plots
plt.figure(figsize=(15, 10))
sns.countplot(data=df, x="transaction_type", palette="Set2")
plt.title("Transaction Type Distribution")
plt.show()
```

```
plt.figure(figsize=(15, 10))
sns.countplot(data=df, x="event_type", palette="Set3")
plt.title("Event Type Distribution")
plt.show()
```

```
<ipython-input-21-8116cfa2c2ef>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.countplot(data=df, x="transaction_type", palette="Set2")
```

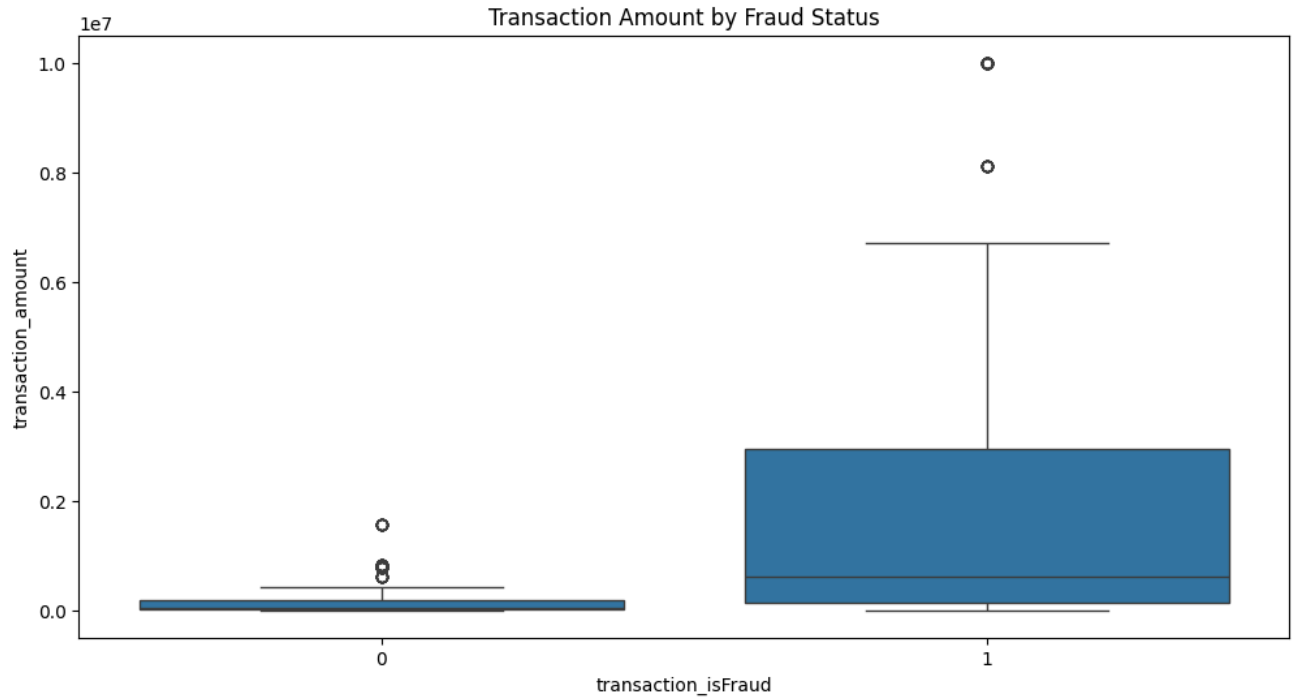



```
<ipython-input-21-8116cfa2c2ef>:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.countplot(data=df, x="event_type", palette="Set3")
```



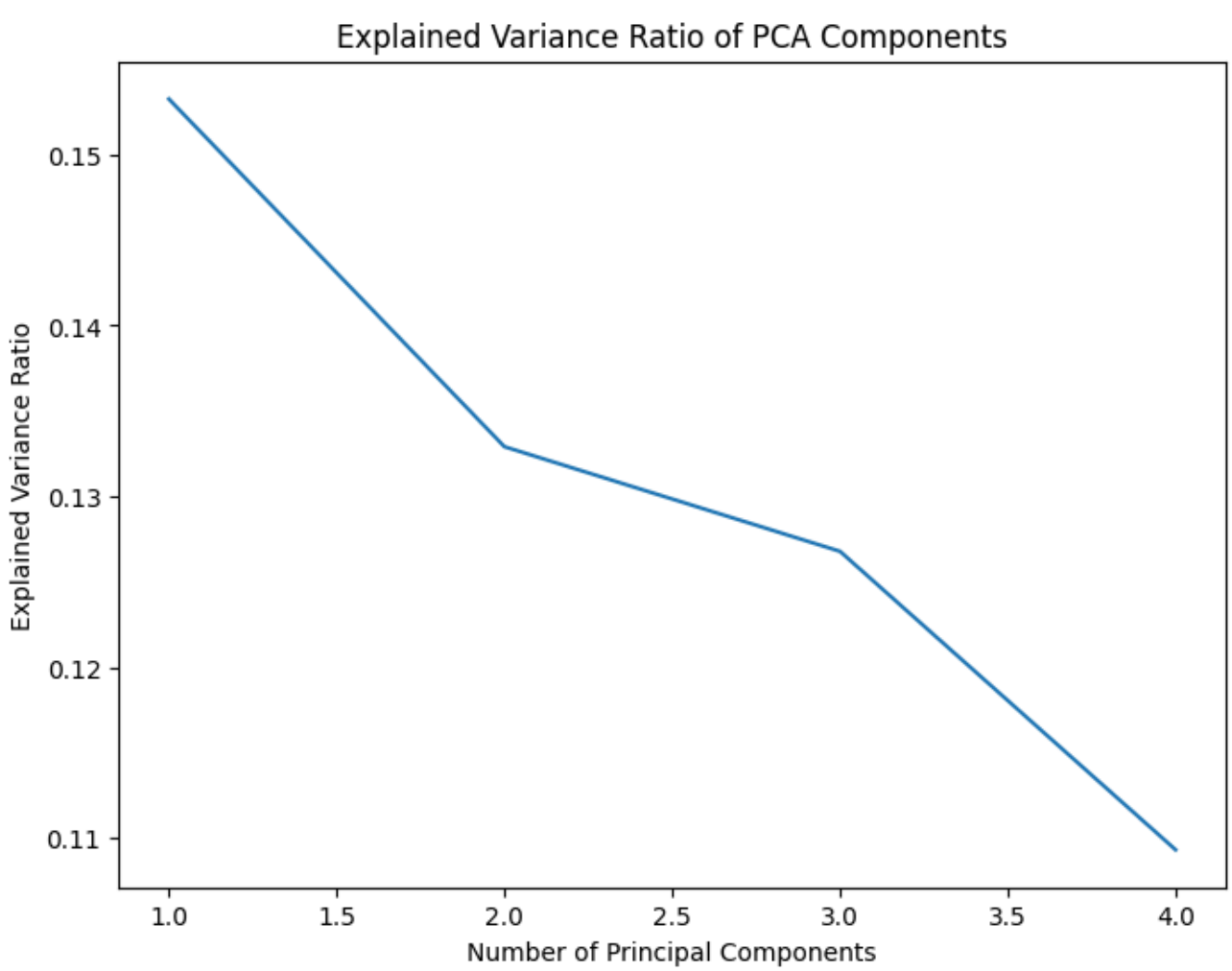

```
# Transaction amount by fraud status
plt.figure(figsize=(12, 6))
```

```
sns.boxplot(x="transaction_isFraud", y="transaction_amount", data=df)
plt.title("Transaction Amount by Fraud Status")
plt.show()

# Event types in fraudulent vs. non-fraudulent transactions
plt.figure(figsize=(15, 6))
sns.countplot(data=df, x="event_type", hue="transaction_isFraud", palette="pastel")
plt.title("Event Types by Fraud Status")
plt.show()
```



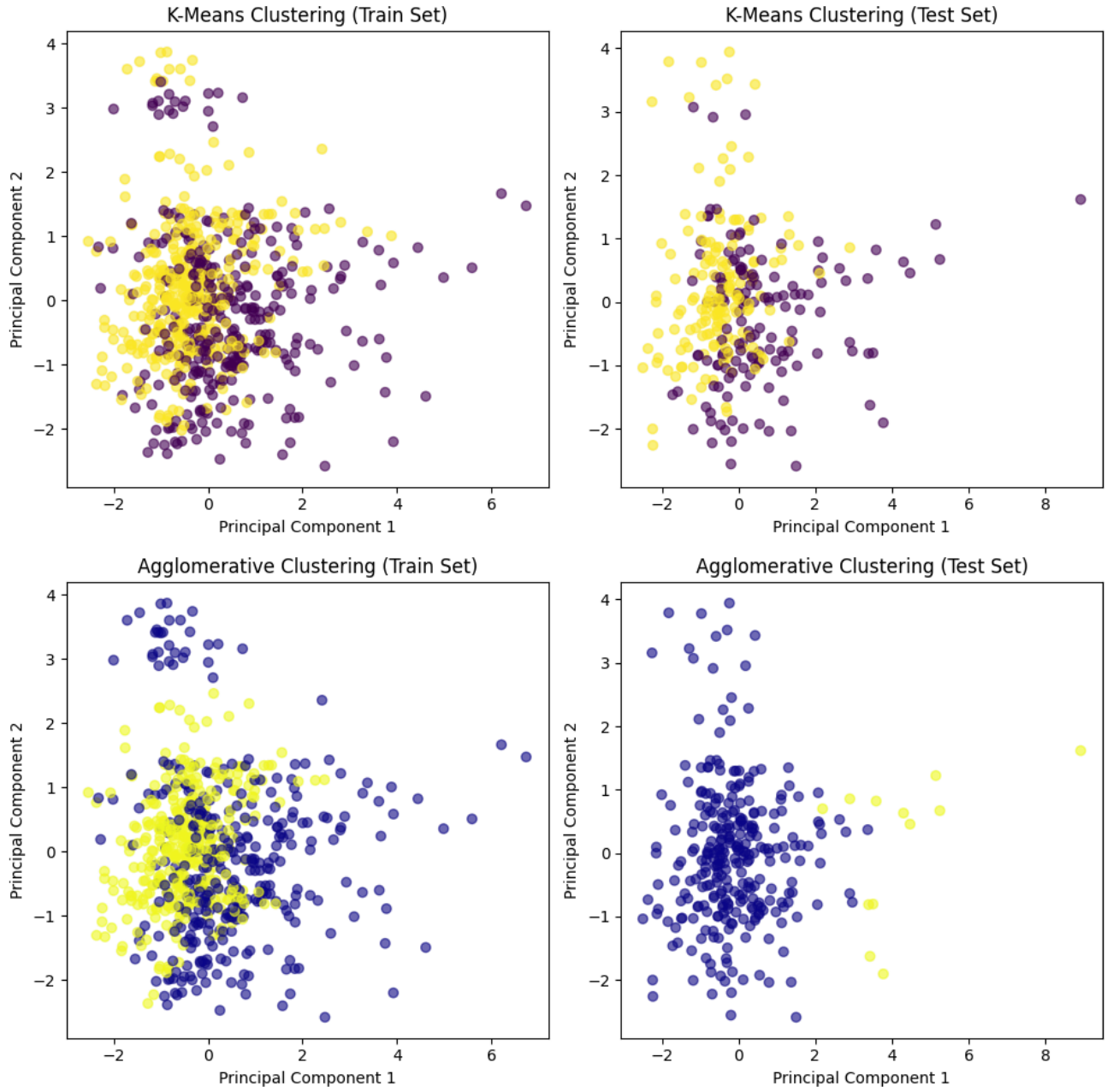

```
# Step 1: Feature Extraction and Encoding
# Encode categorical features
```

```
    for col in categorical_features:
        le = LabelEncoder()
        df[col] = le.fit_transform(df[col])

    # Separate features and target
    X = df[features + categorical_features]
    y = df[target]

    # Step 2: Handle Missing Values
    # Impute missing values using the mean strategy
    imputer = SimpleImputer(strategy="mean")
    X_imputed = imputer.fit_transform(X)

    # Scale the features
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X_imputed)

    # Step 3: Dimensionality Reduction using PCA
    pca = PCA(n_components=4)  # Reduced to 4 components for better generalization
    X_pca = pca.fit_transform(X_scaled)


    # Visualize the explained variance ratio of the PCA components
    plt.figure(figsize=(8, 6))
    plt.plot(range(1, len(pca.explained_variance_ratio_) + 1), pca.explained_variance_ratio_)
    plt.xlabel('Number of Principal Components')
    plt.ylabel('Explained Variance Ratio')
    plt.title('Explained Variance Ratio of PCA Components')
    plt.show()
```




```
# Step 4: Clustering and Neural Network Fraud Score Model
# Split dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_pca, y, test_size=0.3, random_state

# Apply K-Means clustering
kmeans = KMeans(n_clusters=2, random_state=42)
kmeans_labels_train = kmeans.fit_predict(X_train)
kmeans_labels_test = kmeans.predict(X_test)

# Apply Agglomerative Clustering
agg_cluster = AgglomerativeClustering(n_clusters=2)
agg_cluster_labels_train = agg_cluster.fit_predict(X_train)
agg_cluster_labels_test = agg_cluster.fit_predict(X_test)



# Visualize K-Means Clustering on PCA Components
plt.figure(figsize=(10, 5))

# Plot training set with K-Means clustering labels
plt.subplot(1, 2, 1)
plt.scatter(X_train_combined["PC1"], X_train_combined["PC2"], c=kmeans_labels_train, cmap
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
```

```
plt.title('K-Means Clustering (Train Set)')

# Plot testing set with K-Means clustering labels
plt.subplot(1, 2, 2)
plt.scatter(X_test_combined["PC1"], X_test_combined["PC2"], c=kmeans_labels_test, cmap='v
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('K-Means Clustering (Test Set)')

plt.tight_layout()
plt.show()

# Visualize Agglomerative Clustering on PCA Components
plt.figure(figsize=(10, 5))

# Plot training set with Agglomerative Clustering labels
plt.subplot(1, 2, 1)
plt.scatter(X_train_combined["PC1"], X_train_combined["PC2"], c=agg_cluster_labels_train,
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('Agglomerative Clustering (Train Set)')

# Plot testing set with Agglomerative Clustering labels
plt.subplot(1, 2, 2)
plt.scatter(X_test_combined["PC1"], X_test_combined["PC2"], c=agg_cluster_labels_test, cm
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('Agglomerative Clustering (Test Set)')

plt.tight_layout()
plt.show()
```


K-Means Clustering (Train Set)
K-Means Clustering (Test Set)
Agglomerative Clustering (Train Set)
Agglomerative Clustering (Test Set)
Principal Component 1
Principal Component 2

```
# Combine clustering results with PCA components as input for Neural Network
X_train_combined = pd.DataFrame(X_train, columns=[f"PC{i+1}" for i in range(X_train.shape
X_test_combined = pd.DataFrame(X_test, columns=[f"PC{i+1}" for i in range(X_test.shape[1]
X_train_combined["kmeans_cluster"] = kmeans_labels_train
X_train_combined["agg_cluster"] = agg_cluster_labels_train
X_test_combined["kmeans_cluster"] = kmeans_labels_test
X_test_combined["agg_cluster"] = agg_cluster_labels_test

# Step 5: Hyperparameter tuning with GridSearchCV
param_grid = {
    'hidden_layer_sizes': [(10, 5), (50, 25), (50, 25, 10)],
    'activation': ['relu', 'tanh'],
    'learning_rate': ['constant', 'adaptive'],
    'alpha': [0.001, 0.01, 0.1],  # Higher regularization values
    'max_iter': [2000],
    'tol': [1e-4]
}

grid_search = GridSearchCV(MLPClassifier(random_state=42), param_grid, cv=3, scoring='acc
grid_search.fit(X_train_combined, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/numpy/ma/core.py:2820: RuntimeWarning: invali
  _data = np.array(data, dtype=dtype, copy=copy,
```