In [1]:

```
import pandas as pd
```

In [3]:

```
df = pd.read_csv('cust_seg.csv')
df.head(5)
```

```
/var/folders/_0/nmpfpzw134n12j0c0z6jtrw80000gn/T/ipykernel_98731/3036801543.py:1: DtypeWarning: Columns (16) have
mixed types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv('cust_seg.csv')
```

Out[3]:

in_ult1	ind_plan_fin_ult1	ind_pres_fin_ult1	ind_reca_fin_ult1	ind_tjcr_fin_ult1	ind_valo_fin_ult1	ind_viv_fin_ult1	ind_nomina_ult1	ind_nom_pens_ult1	ind_recibo
0	0	0	0	0	0	0	0.0	0.0	
0	0	0	0	0	0	0	0.0	0.0	
0	0	0	0	0	0	0	0.0	0.0	
0	0	0	0	0	0	0	0.0	0.0	
0	0	0	0	0	0	0	0.0	0.0	

In [ ]:

```
import numpy as np
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.feature_selection import SelectKBest, mutual_info_classif
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
import seaborn as sns
import matplotlib.pyplot as plt

# Identify columns with missing values
columns_with_missing = df.columns[df.isnull().any()].tolist()

# Fill missing values in numerical columns with mean
num_imputer = SimpleImputer(strategy='mean')
df[numerical_columns] = num_imputer.fit_transform(df[numerical_columns])

# Fill missing values in categorical columns with mode
cat_imputer = SimpleImputer(strategy='most_frequent')
df[categorical_columns] = cat_imputer.fit_transform(df[categorical_columns])

# Step 2: Outlier Treatment
# Remove outliers using z-score method (example for 'age' column)
z_score = np.abs((df['age'] - df['age'].mean()) / df['age'].std())
df = df[z_score < 3]

# Step 3: Categorical Variable Encoding
# Apply one-hot encoding to categorical variables
encoder = OneHotEncoder()
encoded_features = encoder.fit_transform(df[categorical_columns])
df_encoded = pd.concat([df.drop(categorical_columns, axis=1), pd.DataFrame(encoded_features.toarray())], axis=1)

# Step 4: Imbalanced Dataset
# Split the dataset into features (X) and target variable (y)
X = df_encoded.drop('ind_recibo_ult1', axis=1)
y = df_encoded['ind_recibo_ult1']

# Apply oversampling or undersampling techniques for imbalanced data

# Step 5: Multicollinearity
# Calculate correlation matrix
corr_matrix = df_encoded.corr()

# Use seaborn to visualize the correlation heatmap
sns.heatmap(corr_matrix, cmap='coolwarm', annot=True)
plt.show()

# Apply feature selection techniques or dimensionality reduction

# Step 6: Skewed Distributions
# Apply transformations to reduce skewness (example for 'age' column using logarithmic transformation)
df_encoded['age'] = np.log(df_encoded['age'])

# Step 7: Complex Model Selection
# Split the preprocessed data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train and evaluate a logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

# Step 8: Cross-Validation
# Perform cross-validation and evaluate model performance
scores = cross_val_score(model, X, y, cv=5, scoring='accuracy')

# Print classification report
print(classification_report(y_test, y_pred))
```