```
import os
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.preprocessing import Imputer, LabelEncoder, MinMaxScaler, scale
from sklearn.model_selection import train_test_split, StratifiedShuffleSplit
from sklearn.metrics import confusion_matrix, f1_score, precision_score, recall_score
from sklearn.pipeline import Pipeline, FeatureUnion
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.decomposition import PCA
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.cluster.bicluster import SpectralCoclustering
from sklearn.manifold import TSNE
from sklearn.cluster.bicluster.bicoclustering import BiclusterBoxPlot
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier, VotingClassifier
from sklearn.linear_model.logistic import LogisticRegressionCV
from sklearn.svm.classes import SVC, LinearSVC
from sklearn.neighbors.classification import KNeighborsClassifier
from sklearn.tree.tree import DecisionTreeClassifier
from sklearn.naive_bayes.bernoulli_naive_bayes import BernoulliNB, CategoricalNB
from sklearn.naive_bayes.gaussian_naive_bayes import GaussianNB
from sklearn.naive_bayes.multinomial_naive_bayes import MultinomialNB
from sklearn.metrics.classification import classification_report, accuracy_score
from sklearn.model_selection.search import GridSearchCV
from sklearn.utils import shuffle
from sklearn.cluster.hierarchical.hierarchical_fast import ward_tree
from sklearn.cluster.hierarchical.hierarchical_fast.hierarchical_fast import mst_linkage_core
from sklearn.cluster.hierarchical.hierarchical_fast.hierarchical_fast.cythonized.cython_hierarchical import _hierarchical
from sklearn.cluster.hierarchical.hierarchical_fast.hierarchical_fast.cythonized.cython_hierarchical.cpp.stl_bitset import _stl_bitset
from sklearn.cluster.hierarchical.hierarchical_fast.hierarchical_fast.cythonized.cython_hierarchical.cpp.stl_vector import _stl_vector
from sklearn.cluster.hierarchical.hierarchical_fast.hierarchical_fast.cythonized.cython_hierarchical.cpp.vmath import _vmath
from sklearn.cluster.hierarchical.hierarchical_fast.hierarchical_fast.cythonized.cython_hierarchical.cpp.vmath.fast_exp import _fast_exp
from sklearn.cluster.hierarchical.hierarchical_fast.hierarchical_fast.cythonized.cython_hierarchical.cpp.vmath.fast_log import _fast_log
from sklearn.cluster.hierarchical.hierarchical_fast.hierarchical_fast.cythonized.cython_hierarchical.cpp.vmath.fast_pow import _fast_pow
from sklearn.cluster.hierarchical.hierarchical_fast.hierarchical_fast.cythonized.cython_hierarchical.cpp.vmath.fast_sqrt import _fast_sqrt
from sklearn.cluster.hierarchical.hierarchical_fast.hierarchical_fast.cythonized.cython_hierarchical.cpp.vmath.fast_tanh import _fast_tanh
from sklearn.cluster.hierarchical.hierarchical_fast.hierarchical_fast.cythonized.cython_hierarchical.cpp.vmath.fast_trig import _fast_trig
from sklearn.cluster.hierarchical.hierarchical_fast.hierarchical_fast.cythonized.cython_hierarchical.cpp.vmath.stdmath import _stdmath
from sklearn.cluster.hierarchical.hierarchical_fast.hierarchical_fast.cythonized.cython_hierarchical.cpp.vmath.stdmath.stdmath_fwd import _stdmath_fwd
from sklearn.cluster.hierarchical.hierarchical_fast.hierarchical_fast.cythonized.cython_hierarchical.cpp.vmath.stdmath.stdmath_fwd.pxd import _stdmath_fwd.pxd
from sklearn.cluster.hierarchical.hierarchical_fast.hierarchical_fast.cythonized.cython_hierarchical.cpp.vmath.stdmath.pxd import _stdmath.pxd
from sklearn.cluster.hierarchical.hierarchical_fast.hierarchical_fast.cythonized.cython_hierarchical.cpp.vmath.pxd import _vmath.pxd
from sklearn.cluster.hierarchical.hierarchical_fast.hierarchical_fast.cythonized.cython_hierarchical.pxd import _hierarchical.pxd
from sklearn.cluster.hierarchical.hierarchical_fast.hierarchical_fast.cythonized.cython_hierarchical.pyx import _hierarchical.pyx
from sklearn.cluster.hierarchical import _hierarchical
from sklearn.cluster.hierarchical import _hierarchical_fast
from sklearn.cluster.hierarchical.setup import build_sources
from sklearn.cluster.hierarchical.setup.setup import configuration
from sklearn.cluster.hierarchical.setup.setup.configuration import _get_sources
from sklearn.cluster.hierarchical.setup.setup.configuration.setuptools.extension import Extension
from sklearn.cluster.hierarchical.setup.setup.configuration.setuptools.extension.sysconfig import get_config_var
from sklearn.cluster.hierarchical.setup.setup.configuration.setuptools.extension.sysconfig.distutils.ccompiler import get_default_compiler
from sklearn.cluster.hierarchical.setup.setup.configuration.setuptools.extension.sysconfig.distutils.ccompiler.sysconfig import get_config_var
from sklearn.cluster.hierarchical.setup.setup.configuration.setuptools.extension.sysconfig.distutils.ccompiler.sysconfig.distutils import get_python_inc
from sklearn.cluster.hierarchical.setup.setup.configuration.setuptools.extension.sysconfig.distutils.ccompiler.sysconfig.distutils.sysconfig import get_python_inc
from sklearn.cluster.hierarchical.setup.setup.configuration.setuptools.extension.sysconfig.distutils.ccompiler.sysconfig.distutils.sysconfig.sysconfig import get_config_var
```

Data Preprocessed Importing

```
%cd /kaggle/input/kepler-preprocessed-data/
```

```
/kaggle/input/kepler-preprocessed-data
```

```
data
```

```
Output_exceeded_the_size_limit.
```

	LOCEBUG	NCSBCOMM	NSCCC	OP_BALANCE	BALANCE	OUTSTD	ORDERTO	LOCLEAD	STATEDEST	...	B_1_Avail_0
0	1000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	-0.0
1	1000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	-0.0
2	1000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	-0.0
3	1000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	-0.0
4	1000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	-0.0

5 rows × 181 columns

Data Partition: Cluster Perspectives

```
# Cluster Perspectives
# Partition Data
# Partition Data
# Partition Data
```

```
# ------------------------------------------------------------------
# Load packages
# ------------------------------------------------------------------
library(ggplot2)
"Warning: package 'ggplot2' was built under R version 4.1.3"
"Warning: package 'tibble' was built under R version 4.1.3"
"Warning: package 'tidyr' was built under R version 4.1.3"
"Warning: package 'readr' was built under R version 4.1.3"
"Warning: package 'purrr' was built under R version 4.1.3"
"Warning: package 'dplyr' was built under R version 4.1.3"
"Warning: package 'stringr' was built under R version 4.1.3"
"Warning: package 'forcats' was built under R version 4.1.3"
"Warning: package 'lubridate' was built under R version 4.1.3"
"Warning: package 'timeDate' was built under R version 4.1.3"
"Warning: package 'tidyverse' was built under R version 4.1.3"
"Warning: package 'ggcorrplot' was built under R version 4.1.3"
"Warning: package 'Rtsne' was built under R version 4.1.3"
"Warning: package 'ggfortify' was built under R version 4.1.3"
"Warning: package 'ggrepel' was built under R version 4.1.3"
"Warning: package 'ggpubr' was built under R version 4.1.3"
"Warning: package 'ggsignif' was built under R version 4.1.3"
"Warning: package 'rstatix' was built under R version 4.1.3"
"Warning: package 'Hmisc' was built under R version 4.1.3"
"Warning: package 'survival' was built under R version 4.1.3"
"Warning: package 'Formula' was built under R version 4.1.3"
"Warning: package 'latticeExtra' was built under R version 4.1.3"
"Warning: package 'cluster' was built under R version 4.1.3"
"Warning: package 'rpart' was built under R version 4.1.3"
"Warning: package 'nnet' was built under R version 4.1.3"
"Warning: package 'MASS' was built under R version 4.1.3"
"Warning: package 'ggseqplot' was built under R version 4.1.3"
"Warning: package 'TraMineR' was built under R version 4.1.3"
"Warning: package 'TraMineRextra' was built under R version 4.1.3"
"Warning: package 'ggseqplotly' was built under R version 4.1.3"
"Warning: package 'plotly' was built under R version 4.1.3"
"Warning: package 'ggsankey' was built under R version 4.1.3"
"Warning: package 'ggsankeyfier' was built under R version 4.1.3"
"Warning: package 'ggalluvial' was built under R version 4.1.3"
"Warning: package 'ggparallel' was built under R version 4.1.3"
"Warning: package 'ggpcp' was built under R version 4.1.3"
"Warning: package 'ggbump' was built under R version 4.1.3"
"Warning: package 'ggtreeextra' was built under R version 4.1.3"
"Warning: package 'ggtree
```

```
cm_df_standardized = DataFrame(scale(Matrix(cm_df), dims=1), cm_df.properties)
```

Data set shape is unique - so all variables or little again

Data Normalization

```
cm_df_standardized = DF_ALL[:cm_df[:Sex,:Age,:Job,:Housing,:Saving_accounts,:Checking_accounts,:Purpose,:Credit_amount,:Duration],:]
describe(cm_df_standardized)
```

```
10×7 DataFrame
 Row │ variable  mean    min     median  max     nunique  nmissing  eltype
     │ Symbol    Float64 Float64 Float64 Float64 Nothing  Nothing   DataType
─────┼──────────────────────────────────────────────────────────────────────────
   1 │ Sex       0.7     0.0     1.0     1.0     Nothing  Nothing   Bool
   2 │ Age       0.0     -1.0    -0.2    3.0     Nothing  Nothing   Float64
   3 │ Job       0.9     0.0     1.0     3.0     Nothing  Nothing   Int64
   4 │ Housing   0.7     0.0     1.0     2.0     Nothing  Nothing   Int64
   5 │ Saving_a  0.4     0.0     0.0     3.0     Nothing  Nothing   Int64
   6 │ Checkin   0.4     0.0     0.0     3.0     Nothing  Nothing   Int64
   7 │ Purpose   2.8     0.0     3.0     6.0     Nothing  Nothing   Int64
   8 │ Credit_a  3.3     0.3     2.2     18.4    Nothing  Nothing   Float64
   9 │ Duration  20.9    4.0     18.0    72.0    Nothing  Nothing   Int64
  10 │ Default   0.3     0.0     0.0     1.0     Nothing  Nothing   Int64
```

```
cm_df_standardized[1:10,:]
```

```
10×10 DataFrame
 Row │ Sex  Age     Job  Housing  Saving_a  Checkin  Purpose  Credit_a  Duration  Default
     │ Bool Float64 Int64 Int64    Int64     Int64    Int64    Float64   Int64     Int64
─────┼──────────────────────────────────────────────────────────────────────────────────
   1 │ 1    -0.2    2     1        0         0        5        0.3       6         0
   2 │ 0    -1.0    0     0        1         1        5        1.0       48        1
   3 │ 1    0.8     1     1        2         2        3        2.2       12        0
   4 │ 1    1.6     2     0        0         0        0        3.1       42        0
   5 │ 1    1.6     2     0        0         0        1        4.6       24        1
   6 │ 1    0.0     2     1        0         0        4        1.4       36        0
   7 │ 1    2.4     3     0        0         0        3        7.3       60        1
   8 │ 1    0.8     1     1        2         2        3        2.2       12        0
   9 │ 1    0.8     1     1        2         2        3        2.2       12        0
  10 │ 1    0.8     1     1        2         2        3        2.2       12        0
```

```
cm_df_standardized[:,:Default]
```

```
1000-element Vector{Int64}:
 0
 1
 0
 0
 1
 0
 1
 0
 0
 0
 ⋮
 0
 0
 0
 0
 0
 0
 0
 0
 1
```

```
cm_df_standardized[:,:Default] = convert(Vector{Int}, cm_df_standardized[:,:Default])
```

```
1000-element Vector{Int64}:
 0
 1
 0
 0
 1
 0
 1
 0
 0
 0
 ⋮
 0
 0
 0
 0
 0
 0
 0
 0
 1
```

```
describe(cm_df_standardized, :mean, :std)
```

```
10×3 DataFrame
 Row │ variable  mean    std
     │ Symbol    Float64 Nothing
─────┼──────────────────────────
   1 │ Sex       0.7     Nothing
   2 │ Age       0.0     1.0
   3 │ Job       0.9     0.7
   4 │ Housing   0.7     0.6
   5 │ Saving_a  0.4     0.8
   6 │ Checkin   0.4     0.8
   7 │ Purpose   2.8     1.7
   8 │ Credit_a  3.3     2.8
   9 │ Duration  20.9    12.1
  10 │ Default   0.3     0.5
```

```
In [10]: ts_cube_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1000000 entries, 2010-01-01 00:00:00 to 2013-12-31 23:59:59
Freq: H
Data columns (total 12):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   EU      1000000 non-null  int32
 1   BUS     1000000 non-null  int32
 2   LOT     1000000 non-null  int32
 3   PRT     1000000 non-null  int32
 4   DAY     1000000 non-null  int32
 5   MNTH    1000000 non-null  int32
 6   YEAR    1000000 non-null  int32
 7   hour    1000000 non-null  int32
 8   minute  1000000 non-null  int32
 9   second  1000000 non-null  int32
 10  weekday 1000000 non-null  int32
 11  weekend 1000000 non-null  int32
dtypes: int32(12)
memory usage: 11.4 GB
```

```
In [11]: ts_cube_df[['EU', 'BUS', 'LOT', 'PRT', 'DAY', 'MNTH', 'YEAR', 'hour', 'minute', 'second', 'weekday', 'weekend']].sample(10).T
```

```
Out[11]:
	2011-11-14 11:00:00	2011-11-17 17:00:00	2011-11-29 19:00:00	2011-12-07 21:00:00	2011-12-08 18:00:00	2011-12-09 17:00:00	2011-12-09 18:00:00	2011-12-09 19:00:00	2011-12-09 20:00:00	2011-12-09 21:00:00
EU	0	0	0	0	0	0	0	0	0	0
BUS	0	0	0	0	0	0	0	0	0	0
LOT	0	0	0	0	0	0	0	0	0	0
PRT	0	0	0	0	0	0	0	0	0	0
DAY	14	17	29	7	8	9	9	9	9	9
MNTH	11	11	11	12	12	12	12	12	12	12
YEAR	2011	2011	2011	2011	2011	2011	2011	2011	2011	2011
hour	11	17	19	21	18	17	18	19	20	21
minute	0	0	0	0	0	0	0	0	0	0
second	0	0	0	0	0	0	0	0	0	0
weekday	4	7	2	3	4	5	5	5	5	5
weekend	0	0	0	0	0	0	0	0	0	0
```

```
In [12]: ts_cube_df[['EU', 'BUS', 'LOT', 'PRT', 'DAY', 'MNTH', 'YEAR', 'hour', 'minute', 'second', 'weekday', 'weekend']].sample(10).T
```

```
Out[12]:
	2010-01-01 00:00:00	2010-01-01 01:00:00	2010-01-01 02:00:00	2010-01-01 03:00:00	2010-01-01 04:00:00	2010-01-01 05:00:00	2010-01-01 06:00:00	2010-01-01 07:00:00	2010-01-01 08:00:00	2010-01-01 09:00:00
EU	0	0	0	0	0	0	0	0	0	0
BUS	0	0	0	0	0	0	0	0	0	0
LOT	0	0	0	0	0	0	0	0	0	0
PRT	0	0	0	0	0	0	0	0	0	0
DAY	1	1	1	1	1	1	1	1	1	1
MNTH	1	1	1	1	1	1	1	1	1	1
YEAR	2010	2010	2010	2010	2010	2010	2010	2010	2010	2010
hour	0	1	2	3	4	5	6	7	8	9
minute	0	0	0	0	0	0	0	0	0	0
second	0	0	0	0	0	0	0	0	0	0
weekday	5	5	5	5	5	5	5	5	5	5
weekend	1	1	1	1	1	1	1	1	1	1
```

```
In [13]: ts_cube_df[['EU', 'BUS', 'LOT', 'PRT', 'DAY', 'MNTH', 'YEAR', 'hour', 'minute', 'second', 'weekday', 'weekend']].sample(10).T
```

```
Out[13]:
	2013-12-11 14:00:00	2013-12-11 15:00:00	2013-12-11 16:00:00	2013-12-11 17:00:00	2013-12-11 18:00:00	2013-12-11 19:00:00	2013-12-11 20:00:00	2013-12-11 21:00:00	2013-12-11 22:00:00	2013-12-11 23:00:00
EU	0	0	0	0	0	0	0	0	0	0
BUS	0	0	0	0	0	0	0	0	0	0
LOT	0	0	0	0	0	0	0	0	0	0
PRT	0	0	0	0	0	0	0	0	0	0
DAY	11	11	11	11	11	11	11	11	11	11
MNTH	12	12	12	12	12	12	12	12	12	12
YEAR	2013	2013	2013	2013	2013	2013	2013	2013	2013	2013
hour	14	15	16	17	18	19	20	21	22	23
minute	0	0	0	0	0	0	0	0	0	0
second	0	0	0	0	0	0	0	0	0	0
weekday	3	3	3	3	3	3	3	3	3	3
weekend	0	0	0	0	0	0	0	0	0	0
```

```
In [14]: ts_cube_df[['EU', 'BUS', 'LOT', 'PRT', 'DAY', 'MNTH', 'YEAR', 'hour', 'minute', 'second', 'weekday', 'weekend']].sample(10).T
```

```
Out[14]:
	2010-12-31 00:00:00	2011-01-01 00:00:00	2011-01-01 01:00:00	2011-01-01 02:00:00	2011-01-01 03:00:00	2011-01-01 04:00:00	2011-01-01 05:00:00	2011-01-01 06:00:00	2011-01-01 07:00:00	2011-01-01 08:00:00
EU	0	0	0	0	0	0	0	0	0	0
BUS	0	0	0	0	0	0	0	0	0	0
LOT	0	0	0	0	0	0	0	0	0	0
PRT	0	0	0	0	0	0	0	0	0	0
DAY	31	1	1	1	1	1	1	1	1	1
MNTH	12	1	1	1	1	1	1	1	1	1
YEAR	2010	2011	2011	2011	2011	2011	2011	2011	2011	2011
hour	0	0	1	2	3	4	5	6	7	8
minute	0	0	0	0	0	0	0	0	0	0
second	0	0	0	0	0	0	0	0	0	0
weekday	5	6	6	6	6	6	6	6	6	6
weekend	1	1	1	1	1	1	1	1	1	1
```

```
In [15]: ts_cube_df[['EU', 'BUS', 'LOT', 'PRT', 'DAY', 'MNTH', 'YEAR', 'hour', 'minute', 'second', 'weekday', 'weekend']].sample(10).T
```

```
Out[15]:
	2011-12-31 00:00:00	2012-01-01 00:00:00	2012-01-01 01:00:00	2012-01-01 02:00:00	2012-01-01 03:00:00	2012-01-01 04:00:00	2012-01-01 05:00:00	2012-01-01 06:00:00	2012-01-01 07:00:00	2012-01-01 08:00:00
EU	0	0	0	0	0	0	0	0	0	0
BUS	0	0	0	0	0	0	0	0	0	0
LOT	0	0	0	0	0	0	0	0	0	0
PRT	0	0	0	0	0	0	0	0	0	0
DAY	31	1	1	1	1	1	1	1	1	1
MNTH	12	1	1	1	1	1	1	1	1	1
YEAR	2011	2012	2012	2012	2012	2012	2012	2012	2012	2012
hour	0	0	1	2	3	4	5	6	7	8
minute	0	0	0	0	0	0	0	0	0	0
second	0	0	0	0	0	0	0	0	0	0
weekday	6	7	7	7	7	7	7	7	7	7
weekend	1	1	1	1	1	1	1	1	1	1
```

```
In [16]: ts_cube_df[['EU', 'BUS', 'LOT', 'PRT', 'DAY', 'MNTH', 'YEAR', 'hour', 'minute', 'second', 'weekday', 'weekend']].sample(10).T
```

```
Out[16]:
	2012-12-31 00:00:00	2013-01-01 00:00:00	2013-01-01 01:00:00	2013-01-01 02:00:00	2013-01-01 03:00:00	2013-01-01 04:00:00	2013-01-01 05:00:00	2013-01-01 06:00:00	2013-01-01 07:00:00	2013-01-01 08:00:00
EU	0	0	0	0	0	0	0	0	0	0
BUS	0	0	0	0	0	0	0	0	0	0
LOT	0	0	0	0	0	0	0	0	0	0
PRT	0	0	0	0	0	0	0	0	0	0
DAY	31	1	1	1	1	1	1	1	1	1
MNTH	12	1	1	1	1	1	1	1	1	1
YEAR	2012	2013	2013	2013	2013	2013	2013	2013	2013	2013
hour	0	0	1	2	3	4	5	6	7	8
minute	0	0	0	0	0	0	0	0	0	0
second	0	0	0	0	0	0	0	0	0	0
weekday	7	1	1	1	1	1	1	1	1	1
weekend	1	0	0	0	0	0	0	0	0	0
```

```
In [17]: ts_cube_df.sample(10).T
```

```
Out[17]:
	2011-11-14 11:00:00	2011-11-17 17:00:00	2011-11-29 19:00:00	2011-12-07 21:00:00	2011-12-08 18:00:00	2011-12-09 17:00:00	2011-12-09 18:00:00	2011-12-09 19:00:00	2011-12-09 20:00:00	2011-12-09 21:00:00
EU	0	0	0	0	0	0	0	0	0	0
BUS	0	0	0	0	0	0	0	0	0	0
LOT	0	0	0	0	0	0	0	0	0	0
PRT	0	0	0	0	0	0	0	0	0	0
DAY	14	17	29	7	8	9	9	9	9	9
MNTH	11	11	11	12	12	12	12	12	12	12
YEAR	2011	2011	2011	2011	2011	2011	2011	2011	2011	2011
hour	11	17	19	21	18	17	18	19	20	21
minute	0	0	0	0	0	0	0	0	0	0
second	0	0	0	0	0	0	0	0	0	0
weekday	4	7	2	3	4	5	5	5	5	5
weekend	0	0	0	0	0	0	0	0	0	0
```

```
In [18]: ts_cube_df.sample(10).T
```

```
Out[18]:
	2010-01-01 00:00:00	2010-01-01 01:00:00	2010-01-01 02:00:00	2010-01-01 03:00:00	2010-01-01 04:00:00	2010-01-01 05:00:00	2010-01-01 06:00:00	2010-01-01 07:00:00	2010-01-01 08:00:00	2010-01-01 09:00:00
EU	0	0	0	0	0	0	0	0	0	0
BUS	0	0	0	0	0	0	0	0	0	0
LOT	0	0	0	0	0	0	0	0	0	0
PRT	0	0	0	0	0	0	0	0	0	0
DAY	1	1	1	1	1	1	1	1	1	1
MNTH	1	1	1	1	1	1	1	1	1	1
YEAR	2010	2010	2010	2010	2010	2010	2010	2010	2010	2010
hour	0	1	2	3	4	5	6	7	8	9
minute	0	0	0	0	0	0	0	0	0	0
second	0	0	0	0	0	0	0	0	0	0
weekday	5	5	5	5	5	5	5	5	5	5
weekend	1	1	1	1	1	1	1	1	1	1
```

```
In [19]: ts_cube_df.sample(10).T
```

```
Out[19]:
	2013-12-11 14:00:00	2013-12-11 15:00:00	2013-12-11 16:00:00	2013-12-11 17:00:00	2013-12-11 18:00:00	2013-12-11 19:00:00	2013-12-11 20:00:00	2013-12-11 21:00:00	2013-12-11 22:00:00	2013-12-11 23:00:00
EU	0	0	0	0	0	0	0	0	0	0
BUS	0	0	0	0	0	0	0	0	0	0
LOT	0	0	0	0	0	0	0	0	0	0
PRT	0	0	0	0	0	0	0	0	0	0
DAY	11	11	11	11	11	11	11	11	11	11
MNTH	12	12	12	12	12	12	12	12	12	12
YEAR	2013	2013	2013	2013	2013	2013	2013	2013	2013	2013
hour	14	15	16	17	18	19	20	21	22	23
minute	0	0	0	0	0	0	0	0	0	0
second	0	0	0	0	0	0	0	0	0	0
weekday	3	3	3	3	3	3	3	3	3	3
weekend	0	0	0	0	0	0	0	0	0	0
```

```
In [20]: ts_cube_df.sample(10).T
```

```
Out[20]:
	2010-12-31 00:00:00	2011-01-01 00:00:00	2011-01-01 01:00:00	2011-01-01 02:00:00	2011-01-01 03:00:00	2011-01-01 04:00:00	2011-01-01 05:00:00	2011-01-01 06:00:00	2011-01-01 07:00:00	2011-01-01 08:00:00
EU	0	0	0	0	0	0	0	0	0	0
BUS	0	0	0	0	0	0	0	0	0	0
LOT	0	0	0	0	0	0	0	0	0	0
PRT	0	0	0	0	0	0	0	0	0	0
DAY	31	1	1	1	1	1	1	1	1	1
MNTH	12	1	1	1	1	1	1	1	1	1
YEAR	2010	2011	2011	2011	2011	2011	2011	2011	2011	2011
hour	0	0	1	2	3	4	5	6	7	8
minute	0	0	0	0	0	0	0	0	0	0
second	0	0	0	0	0	0	0	0	0	0
weekday	6	6	6	6	6	6	6	6	6	6
weekend	1	1	1	1	1	1	1	1	1	1
```

```
In [21]: ts_cube_df.sample(10).T
```

```
Out[21]:
	2011-12-31 00:00:00	2012-01-01 00:00:00	2012-01-01 01:00:00	2012-01-01 02:00:00	2012-01-01 03:00:00	2012-01-01 04:00:00	2012-01-01 05:00:00	2012-01-01 06:00:00	2012-01-01 07:00:00	2012-01-01 08:00:00
EU	0	0	0	0	0	0	0	0	0	0
BUS	0	0	0	0	0	0	0	0	0	0
LOT	0	0	0	0	0	0	0	0	0	0
PRT	0	0	0	0	0	0	0	0	0	0
DAY	31	1	1	1	1	1	1	1	1	1
MNTH	12	1	1	1	1	1	1	1	1	1
YEAR	2011	2012	2012	2012	2012	2012	2012	2012	2012	2012
hour	0	0	1	2	3	4	5	6	7	8
minute	0	0	0	0	0	0	0	0	0	0
second	0	0	0	0	0	0	0	0	0	0
weekday	6	7	7	7	7	7	7	7	7	7
weekend	1	1	1	1	1	1	1	1	1	1
```

```
In [22]: ts_cube_df.sample(10).T
```

```
Out[22]:
	2012-12-31 00:00:00	2013-01-01 00:00:00	2013-01-01 01:00:00	2013-01-01 02:00:00	2013-01-01 03:00:00	2013-01-01 04:00:00	2013-01-01 05:00:00	2013-01-01 06:00:00	2013-01-01 07:00:00	2013-01-01 08:00:00
EU	0	0	0	0	0	0	0	0	0	0
BUS	0	0	0	0	0	0	0	0	0	0
LOT	0	0	0	0	0	0	0	0	0	0
PRT	0	0	0	0	0	0	0	0	0	0
DAY	31	1	1	1	1	1	1	1	1	1
MNTH	12	1	1	1	1	1	1	1	1	1
YEAR	2012	2013	2013	2013	2013	2013	2013	2013	2013	2013
hour	0	0	1	2	3	4	5	6	7	8
minute	0	0	0	0	0	0	0	0	0	0
second	0	0	0	0	0	0	0	0	0	0
weekday	7	1	1	1	1	1	1	1	1	1
weekend	1	0	0	0	0	0	0	0	0	0
```

```
In [23]: ts_cube_df.sample(10).T
```

```
Out[23]:
	2010-01-01 00:00:00	2010-01-01 01:00:00	2010-01-01 02:00:00	2010-01-01 03:00:00	2010-01-01 04:00:00	2010-01-01 05:00:00	2010-01-01 06:00:00	2010-01-01 07:00:00	2010-01-01 08:00:00	2010-01-01 09:00:00
EU	0	0	0	0	0	0	0	0	0	0
BUS	0	0	0	0	0	0	0	0	0	0
LOT	0	0	0	0	0	0	0	0	0	0
PRT	0	0	0	0	0	0	0	0	0	0
DAY	1	1	1	1	1	1	1	1	1	1
MNTH	1	1	1	1	1	1	1	1	1	1
YEAR	2010	2010	2010	2010	2010	2010	2010	2010	2010	2010
hour	0	1	2	3	4	5	6	7	8	9
minute	0	0	0	0	0	0	0	0	0	0
second	0	0	0	0	0	0	0	0	0	0
weekday	5	5	5	5	5	5	5	5	5	5
weekend	1	1	1	1	1	1	1	1	1	1
```

```
In [24]: ts_cube_df.sample(10).T
```

```
Out[24]:
	2013-12-11 14:00:00	2013-12-11 15:00:00	2013-12-11 16:00:00	2013-12-11 17:00:00	2013-12-11 18:00:00	2013-12-11 19:00:00	2013-12-11 20:00:00	2013-12-11 21:00:00	2013-12-11 22:00:00	2013-12-11 23:00:00
EU	0	0	0	0	0	0	0	0	0	0
BUS	0	0	0	0	0	0	0	0	0	0
LOT	0	0	0	0	0	0	0	0	0	0
PRT	0	0	0	0	0	0	0	0	0	0
DAY	11	11	11	11	11	11	11	11	11	11
MNTH	12	12	12	12	12	12	12	12	12	12
YEAR	2013	2013	2013	2013	2013	2013	2013	2013	2013	2013
hour	14	15	16	17	18	19	20	21	22	23
minute	0	0	0	0	0	0	0	0	0	0
second	0	0	0	0	0	0	0	0	0	0
weekday	3	3	3	3	3	3	3	3	3	3
weekend	0	0	0	0	0	0	0	0	0	0
```

```
In [25]: ts_cube_df.sample(10).T
```

```
Out[25]:
	2010-12-31 00:00:00	2011-01-01 00:00:00	2011-01-01 01:00:00	2011-01-01 02:00:00	2011-01-01 03:00:00	2011-01-01 04:00:00	2011-01-01 05:00:00	2011-01-01 06:00:00	2011-01-01 07:00:00	2011-01-01 08:00:00
EU	0	0	0	0	0	0	0	0	0	0
BUS	0	0	0	0	0	0	0	0	0	0
LOT	0	0	0	0	0	0	0	0	0	0
PRT	0	0	0	0	0	0	0	0	0	0
DAY	31	1	1	1	1	1	1	1	1	1
MNTH	12	1	1	1	1	1	1	1	1	1
YEAR	2010	2011	2011	2011	2011	2011	2011	2011	2011	2011
hour	0	0	1	2	3	4	5	6	7	8
minute	0	0	0	0	0	0	0	0	0	0
second	0	0	0	0	0	0	0	0	0	0
weekday	6	6	6	6	6	6	6	6	6	6
weekend	1	1	1	1	1	1	1	1	1	1
```

```
In [26]: ts_cube_df.sample(10).T
```

```
Out[26]:
	2011-12-31 00:00:00	2012-01-01 00:00:00	2012-01-01 01:00:00	2012-01-01 02:00:00	2012-01-01 03:00:00	2012-01-01 04:00:00	2012-01-01 05:00:00	2012-01-01 06:00:00	2012-01-01 07:00:00	2012-01-01 08:00:00
EU	0	0	0	0	0	0	0	0	0	0
BUS	0	0	0	0	0	0	0	0	0	0
LOT	0	0	0	0	0	0	0	0	0	0
PRT	0	0	0	0	0	0	0	0	0	0
DAY	31	1	1	1	1	1	1	1	1	1
MNTH	12	1	1	1	1	1	1	1	1	1
YEAR	2011	2012	2012	2012	2012	2012	2012	2012	2012	2012
hour	0	0	1	2	3	4	5	6	7	8
minute	0	0	0	0	0	0	0	0	0	0
second	0	0	0	0	0	0	0	0	0	0
weekday	6	7	7	7	7	7	7	7	7	7
weekend	1	1	1	1	1	1	1	1	1	1
```

```
In [27]: ts_cube_df.sample(10).T
```

```
Out[27]:
	2012-12-31 00:00:00	2013-01-01 00:00:00	2013-01-01 01:00:00	2013-01-01 02:00:00	2013-01-01 03:00:00	2013-01-01 04:00:00	2013-01-01 05:00:00	2013-01-01 06:00:00	2013-01-01 07:00:00	2013-01-01 08:00:00
EU	0	0	0	0	0	0	0	0	0	0
BUS	0	0	0	0	0	0	0	0	0	0
LOT	0	0	0	0	0	0	0	0	0	0
PRT	0	0	0	0	0	0	0	0	0	0
DAY	31	1	1	1	1	1	1	1	1	1
MNTH	12	1	1	1	1	1	1	1	1	1
YEAR	2012	2013	2013	2013	2013	2013	2013	2013	2013	2013
hour	0	0	1	2	3	4	5	6	7	8
minute	0	0	0	0	0	0	0	0	0	0
second	0	0	0	0	0	0	0	0	0	0
weekday	7	1	1	1	1	1	1	1	1	1
weekend	1	0	0	0	0	0	0	0	0	0
```

```
In [28]: ts_cube_df.sample(10).T
```

```
Out[28]:
	2010-01-01 00:00:00	2010-01-01 01:00:00	2010-01-01 02:00:00	2010-01-01 03:00:00	2010-01-01 04:00:00	2010-01-01 05:00:00	2010-01-01 06:00:00	2010-01-01 07:00:00	2010-01-01 08:00:00	2010-01-01 09:00:00
EU	0	0	0	0	0	0	0	0	0	0
BUS	0	0	0	0	0	0	0	0	0	0
LOT	0	0	0	0	0	0	0	0	0	0
PRT	0	0	0	0	0	0	0	0	0	0
DAY	1	1	1	1	1	1	1	1	1	1
MNTH	1	1	1	1	1	1	1	1	1	1
YEAR	2010	2010	2010	2010	2010	2010	2010	2010	2010	2010
hour	0	1	2	3	4	5	6	7	8	9
minute	0	0	0	0	0	0	0	0	0	0
second	0	0	0	0	0	0	0	0	0	0
weekday	5	5	5	5	5	5	5	5	5	5
weekend	1	1	1	1	1	1	1	1	1	1
```

```
In [29]: ts_cube_df.sample(10).T
```

```
Out[29]:
	2013-12-11 14:00:00	2013-12-11 15:00:00	2013-12-11 16:00:00	2013-12-11 17:00:00	2013-12-11 18:00:00	2013-12-11 19:00:00	2013-12-11 20:00:00	2013-12-11 21:00:00	2013-12-11 22:00:00	2013-12-11 23:00:00
EU	0	0	0	0	0	0	0	0	0	0
BUS	0	0	0	0	0	0	0	0	0	0
LOT	0	0	0	0	0	0	0	0	0	0
PRT	0	0	0	0	0	0	0	0	0	0
DAY	11	11	11	11	11	11	11	11	11	11
MNTH	12	12	12	12	12	12	12	12	12	12
YEAR	2013	2013	2013	2013	2013	2013	2013	2013	2013	2013
hour	14	15	16	17	18	19	20	21	22	23
minute	0	0	0	0	0	0	0	0	0	0
second	0	0	0	0	0	0	0	0	0	0
weekday	3	3	3	3	3	3	3	3	3	3
weekend	0	0	0	0	0	0	0	0	0	0
```

```
In [30]: ts_cube_df.sample(10).T
```

```
Out[30]:
	2010-12-31 00:00:00	2011-01-01 00:00:00	2011-01-01 01:00:00	2011-01-01 02:00:00	2011-01-01 03:00:00	2011-01-01 04:00:00	2011-01-01 05:00:00	2011-01-01 06:00:00	2011-01-01 07:00:00	2011-01-01 08:00:00
EU	0	0	0	0	0	0	0	0	0	0
BUS	0	0	0	0	0	0	0	0	0	0
LOT	0	0	0	0	0	0	0	0	0	0
PRT	0	0	0	0	0	0	0	0	0	0
DAY	31	1	1	1	1	1	1	1	1	1
MNTH	12	1	1	1	1	1	1	1	1	1
YEAR	2010	2011	2011	2011	2011	2011	2011	2011	2011	2011
hour	0	0	1	2	3	4	5	6	7	8
minute	0	0	0	0	0	0	0	0	0	0
second	0	0	0	0	0	0	0	0	0	0
weekday	6	6	6	6	6	6	6	6	6	6
weekend	1	1	1	1	1	1	1	1	1	1
```

```
In [31]: ts_cube_df.sample(10).T
```

```
Out[31]:
	2011-12-31 00:00:00	2012-01-01 00:00:00	2012-01-01 01:00:00	2012-01-01 02:00:00	2012-01-01 03:00:00	2012-01-01 04:00:00	2012-01-01 05:00:00	2012-01-01 06:00:00	2012-01-01 07:00:00	2012-01-01 08:00:00
EU	0	0	0	0	0	0	0	0	0	0
BUS	0	0	0	0	0	0	0	0	0	0
LOT	0	0	0	0	0	0	0	0	0	0
PRT	0	0	0	0	0	0	0	0	0	0
DAY	31	1	1	1	1	1	1	1	1	1
MNTH	12	1	1	1	1	1	1	1	1	1
YEAR	2011	2012	2012	2012	2012	2012	2012	2012	2012	2012
hour	0	0	1	2	3	4	5	6	7	8
minute	0	0	0	0	0	0	0	0	0	0
second	0	0	0	0	0	0	0	0	0	0
weekday	6	7	7	7	7	7	7	7	7	7
weekend	1	1	1	1	1	1	1	1	1	1
```

```
In [32]: ts_cube_df.sample(10).T
```

```
Out[32]:
	2012-12-31 00:00:00	2013-01-01 00:00:00	2013-01-01 01:00:00	2013-01-01 02:00:00	2013-01-01 03:00:00	2013-01-01 04:00:00	2013-01-01 05:00:00	2013-01-01 06:00:00	2013-01-01 07:00:00	2013-01-01 08:00:00
EU	0	0	0	0	0	0	0	0	0	0
BUS	0	0	0	0	0	0	0	0	0	0
LOT	0	0	0	0	0	0	0	0	0	0
PRT	0	0	0	0	0	0	0	0	0	0
DAY	31	1	1	1	1	1	1	1	1	1
MNTH	12	1	1	1	1	1	1	1	1	1
YEAR	2012	2013	2013	2013	2013	2013	2013	2013	2013	2013
hour	0	0	1	2	3	4	5	6	7	8
minute	0	0	0	0	0	0	0	0	0	0
second	0	0	0	0	0	0	0	0	0	0
weekday	7	1	1	1	1	1	1	1	1	1
weekend	1	0	0	0	0	0	0	0	0	0
```

```
In [33]: ts_cube_df.sample(10).T
```

```
Out[33]:
	2010-01-01 00:00:00	2010-01-01 01:00:00	2010-01-01 02:00:00	2010-01-01 03:00:00	2010-01-01 04:00:00	2010-01-01 05:00:00	2010-01-01 06:00:00	2010-01-01 07:00:00	2010-01-01 08:00:00	2010-01-01 09:00:00
EU	0	0	0	0	0	0	0	0	0	0
BUS	0	0	0	0	0	0	0	0	0	0
LOT	0	0	0	0	0	0	0	0	0	0
PRT	0	0	0	0	0	0	0	0	0	0
DAY	1	1	1	1	1	1	1	1	1	1
MNTH	1	1	1	1	1	1	1	1	1	1
YEAR	2010	2010	2010	2010	2010	2010	2010	2010	2010	2010
hour	0	1	2	3	4	5	6	7	8	9
minute	0	0	0	0	0	0	0	0	0	0
second	0	0	0	0	0	0	0	0	0	0
weekday	5	5	5	5	5	5	5	5	5	5
weekend	1	1	1	1	1	1	1	1	1	1
```

```
In [34]: ts_cube_df.sample(10).T
```

```
Out[34]:
	2013-12-11 14:00:00	2013-12-11 15:00:00	2013-12-11 16:00:00	2013-12-11 17:00:00	2013-12-11 18:00:00	2013-12-11 19:00:00	2013-12-11 20:00:00	2013-12-11 21:00:00	2013-12-11 22:00:00	2013-12-11 23:00:00
EU	0	0	0	0	0	0	0	0	0	0
BUS	0	0	0	0	0	0	0	0	0	0
LOT	0	0	0	0	0	0	0	0	0	0
PRT	0	0	0	0	0	0	0	0	0	0
DAY	11	11	11	11	11	11	11	11	11	11
MNTH	12	12	12	12	12	12	12	12	12	12
YEAR	2013	2013	2013	2013	2013	2013	2013	2013	2013	2013
hour	14	15	16	17	18	19	20	21	22	23
minute	0	0	0	0	0	0	0	0	0	0
second	0	0	0	0	0	0	0	0	0	0
weekday	3	3	3	3	3	3	3	3	3	3
weekend	0	0	0	0	0	0	0	0	0	0
```

```
In [35]: ts_cube_df.sample(10).T
```

```
Out[35]:
	2010-12-31 00:00:00	2011-01-01 00:00:00	2011-01-01 01:00:00	2011-01-01 02:00:00	2011-01-01 03:00:00	2011-01-01 04:00:00	2011-01-01 05:00:00	2011-01-01 06:00:00	2011-01-01 07:00:00	2011-01-01 08:00:00
EU	0	0	0	0	0	0	0	0	0	0
BUS	0	0	0	0	0	0	0	0	0	0
LOT	0	0	0	0	0	0	0	0	0	0
PRT	0	0	0	0	0	0	0	0	0	0
DAY	31	1	1	1	1	1	1	1	1	1
MNTH	12	1	1	1	1	1	1	1	1	1
YEAR	2010	2011	2011	2011	2011	2011	2011	2011	2011	2011
hour	0	0	1	2	3	4	5	6	7	8
minute	0	0	0	0	0	0	0	0	0	0
second	0	0	0	0	0	0	0	0	0	0
weekday	6	6	6	6	6	6	6	6	6	6
weekend	1	1	1	1	1	1	1	1	1	1
```

```
In [36]: ts_cube_df.sample(10).T
```

```
Out[36]:
	2011-12-31 00:00:00	2012-01-01 00:00:00	2012-01-01 01:00:00	2012-01-01 02:00:00	2012-01-01 03:00:00	2012-01-01 04:00:00	2012-01-01 05:00:00	2012-01-01 06:00:00	2012-01-01 07:00:00	2012-01-01 08:00:00
EU	0	0	0	0	0	0	0	0	0	0
BUS	0	0	0	0	0	0	0	0	0	0
LOT	0	0	0	0	0	0	0	0	0	0
PRT	0	0	0	0	0	0	0	0	0	0
DAY	31	1	1	1	1	1	1	1	1	1
MNTH	12	1	1	1	1	1	1	1	1	1
YEAR	2011	2012	2012	2012	2012	2012	2012	2012	2012	2012
hour	0	0	1	2	3	4	5	6	7	8
minute	0	0	0	0	0	0	0	0	0	0
second	0	0	0	0	0	0	0	0	0	0
weekday	6	7	7	7	7	7	7	7	7	7
weekend	1	1	1	1	1	1	1	1	1	1
```

```
In [37]: ts_cube_df.sample(10).T
```

```
Out[37]:
	2012-12-31 00:00:00	2013-01-01 00:00:00	2013-01-01 01:00:00	2013-01-01 02:00:00	2013-01-01 03:00:00	2013-01-01 04:00:00	2013-01-01 05:00:00	2013-01-01 06:00:00	2013-01-01 07:00:00	2013-01-01 08:00:00
EU	0	0	0	0	0	0	0	0	0	0
BUS	0	0	0	0	0	0	0	0	0	0
LOT	0	0	0	0	0	0	0	0	0	0
PRT	0	0	0	0	0	0	0	0	0	0
DAY	31	1	1	1	1	1	1	1	1	1
MNTH	12	1	1	1	1	1	1	1	1	1
YEAR	2012	2013	2013	2013	2013	2013	2013	2013	2013	2013
hour	0	0	1	2	3	4	5	6	7	8
minute	0	0	0	0	0	0	0	0	0	0
second	0	0	0	0	0	0	0	0	0	0
weekday	7	1	1	1	1	1	1	1	1	1
weekend	1	0	0	0	0	0	0	0	0	0
```

```
In [38]: ts_cube_df.sample(10).T
```

```
Out[38]:
	2010-01-01 00:00:00	2010-01-01 01:00:00	2010-01-01 02:00:00	2010-01-01 03:00:00	2010-01-01 04:00:00	2010-01-01 05:00:00	2010-01-01 06:00:00	2010-01-01 07:00:00	2010-01-01 08:00:00	2010-01-01 09:00:00
EU	0	0	0	0	0	0	0	0	0	0
BUS	0	0	0	0	0	0	0	0	0	0
LOT	0	0	0	0	0	0	0	0	0	0
PRT	0	0	0	0	0	0	0	0	0	0
DAY	1	1	1	1	1	1	1	1	1	1
MNTH	1	1	1	1	1	1	1	1	1	1
YEAR	2010	2010	2010	2010	2010	2010	2010	2010	2010	2010
hour	0	1	2	3	4	5	6	7	8	9
minute	0	0	0	0	0	0	0	0	0	0
second	0	0	0	0	0	0	0	0	0	0
weekday	5	5	5	5	5	5	5	5	5	5
weekend	1	1	1	1	1	1	1	1	1	1
```

```
In [39]: ts_cube_df.sample(10).T
```

```
Out[39]:
	2013-12-11 14:00:00	2013-12-11 15:00:00	2013-12-11 16:00:00	2013-12-11 17:00:00	2013-12-11 18:00:00	2013-12-11 19:00:00	2013-12-11 20:00:00	2013-12-11 21:00:00	2013-12-11 22:00:00	2013-12-11 23:00:00
EU	0	0	0	0	0	0	0	0	0	0
BUS	0	0	0	0	0	0	0	0	0	0
LOT	0	0	0	0	0	0	0	0	0	0
PRT	0	0	0	0	0	0	0	0	0	0
DAY	11	11	11	11	11	11	11	11	11	11
MNTH	12	12	12	12	12	12	12	12	12	12
YEAR	2013	2013	2013	2013	2013	2013	2013	2013	2013	2013
hour	14	15	16	17	18	19	20	21	22	23
minute	0	0	0	0	0	0	0	0	0	0
second	0	0	0	0	0	0	0	0	0	0
weekday	3	3	3	3	3	3	3	3	3	3
weekend	0	0	0	0	0	0	0	0	0	0
```

```
In [40]: ts_cube_df.sample(10).T
```

```
Out[40]:
	2010-12-31 00:00:00	2011-01-01 00:00:00	2011-01-01 01:00:00	2011-01-01 02:00:00	2011-01-01 03:00:00	2011-01-01 04:00:00	2011-01-01 05:00:00	2011-01-01 06:00:00	2011-01-01 07:00:00	2011-01-01 08:00:00
EU	0	0	0	0	0	0	0	0	0	0
BUS	0	0	0	0	0	0	0	0	0	0
LOT	0	0	0	0	0	0	0	0	0	0
PRT	0	0	0	0	0	0	0	0	0	0
DAY	31	1	1	1	1	1	1	1	1	1
MNTH	12	1	1	1	1	1	1	1	1	1
YEAR	2010	2011	2011	2011	2011	2011	2011	2011	2011	2011
hour	0	0	1	2	3	4	5	6	7	8
minute	0	0	0	0	0	0	0	0	0	0
second	0	0	0	0	0	0	0	0	0	0
weekday	6	6	6	6	6	6	6	6	6	6
weekend	1	1	1	1	1	1	1	1	1	1
```

```
In [41]: ts_cube_df.sample(10).T
```

```
Out[41]:
	2011-12-31 00:00:00	2012-01-01 00:00:00	2012-01-01 01:00:00	2012-01-01 02:00:00	2012-01-01 03:00:00	2012-01-01 04:00:00	2012-01-01 05:00:00	2012-01-01 06:00:00	2012-01-01 07:00:00	2012-01-01 08:00:00
EU	0	0	0	0	0	0	0	0	0	0
BUS	0	0	0	0	0	0	0	0	0	0
LOT	0	0	0	0	0	0	0	0	0	0
PRT	0	0	0	0	0	0	0	0	0	0
DAY	31	1	1	1	1	1	1	1	1	1
MNTH	12	1	1	1	1	1	1	1	1	1
YEAR	2011	2012	2012	2012	2012	2012	2012	2012	2012	2012
hour	0	0	1	2	3	4	5	6	7	8
minute	0	0	0	0	0	0	0	0	0	0
second	0	0	0	0	0	0	0	0	0	0
weekday	6	7	7	7	7	7	7	7	7	7
weekend	1	1	1	1	1	1	1	1	1	1
```

```
In [42]: ts_cube_df.sample(10).T
```

```
Out[42]:
	2012-12-31 00:00:00	2013-01-01 00:00:00	2013-01-01 01:00:00	2013-01-01 02:00:00	2013-01-01 03:00:00	2013-01-01 04:00:00	2013-01-01 05:00:00	2013-01-01 06:00:00	2013-01-01 07:00:00	2013-01-01 08:00:00
EU	0	0	0	0	0	0	0	0	0	0
BUS	0	0	0	0	0	0	0	0	0	0
LOT	0	0	0	0	0	0	0	0	0	0
PRT	0	0	0	0	0	0	0	0	0	0
DAY	31	1	1	1	1	1	1	1	1	1
MNTH	12	1	1	1	1	1	1	1	1	1
YEAR	2012	2013	2013	2013	2013	2013	2013	2013	2013	2013
hour	0	0	1	2	3	4	5	6	7	8
minute	0	0	0	0	0	0	0	0	0	0
second	0	0	0	0	0	0	0	0	0	0
weekday	7	1	1	1	1	1	1	1	1	1
weekend	1	0	0	0	0	0	0	0	0	0
```

```
In [43]: ts_cube_df.sample(10).T
```

```
Out[43]:
	2010-01-01 00:00:00	2010-01-01 01:00:00	2010-01-01 02:00:00	2010-01-01 03:00:00	2010-01-01 04:00:00	2010-01-01 05:00:00	2010-01-01 06:00:00	2010-01-01 07:00:00	2010-01-01 08:00:00	2010-01-01 09:00:00
EU	0	0	0	0	0	0	0	0	0	0
BUS	0	0	0	0	0	0	0	0	0	0
LOT	0	0	0	0	0	0	0	0	0	0
PRT	0	0	0	0	0	0	0	0	0	0
DAY	1	1	1	1	1	1	1	1	1	1
MNTH	1	1	1	1	1	1	1	1	1	1
YEAR	2010	2010	2010	2010	2010	2010	2010	2010	2010	2010
hour	0	1	2	3	4	5	6	7	8	9
minute	0	0	0	0	0	0	0	0	0	0
second	0	0	0	0	0	0	0	0	0	0
weekday	5	5	5	5	5	5	5	5	5	5
weekend	1	1	1	1	
```

```
# Calculate clusters
cut_1 <- cutree(hclust_1, k=4)$"Pineapple","Pineapple","Pineapple")
# Plot the dendrogram with colored groups and rectangles around clusters
plot(hclust_1, main="Hierarchical Clustering Dendrogram", xlab="", sub="", cex=0.9)
rect.hclust(hclust_1, k=4, which=c(1:4), border=c("red", "blue", "green", "purple"))
# Add legend
legend("topright", legend=c("Cluster 1", "Cluster 2", "Cluster 3", "Cluster 4"), fill=c("red", "blue", "green", "purple"), cex=0.8)
# Calculate cophenetic correlation coefficient
cophenetic_cor <- cophenetic(hclust_1)
cor_coef <- cor(dist_1, cophenetic_cor)
print("Cophenetic correlation coefficient:", round(cor_coef, 3))
# Alternative method using dendextend for more customization
library(dendextend)
dend <- as.dendrogram(hclust_1)
dend <- color_branches(dend, k=4, col=c("red", "blue", "green", "purple"))
plot(dend, main="Hierarchical Clustering Dendrogram", xlab="", sub="")
```

```
## [1] "Cophenetic correlation coefficient:" 0.739
```

```
# Create correlation matrix for variables
cor_matrix <- cor(cluster_data[, c("Pineapple", "Pineapple", "Pineapple")])
# Create heatmap
library(pheatmap)
pheatmap(cor_matrix, main="Correlation Matrix", display_numbers=TRUE)
```

```
# Create PCA biplot
library(ggfactoextra)
fviz_pca_biplot(pca_1, geom.ind="point", pointsize=2, palette=c("red", "blue", "green", "purple"), addEllipses=TRUE, ellipse.level=0.95, legend.title="Cluster")
```

```
# Create scree plot
library(factoextra)
fviz_eig(pca_1, addlabels=TRUE, barfill="steelblue", barcolor="darkblue")
```

```
# Create biplot with PC1 and PC2
library(ggfactoextra)
fviz_pca_biplot(pca_1, geom.ind="point", pointsize=2, palette=c("red", "blue", "green", "purple"), addEllipses=TRUE, ellipse.level=0.95, legend.title="Cluster")
```

```
# Create scree plot
library(factoextra)
fviz_eig(pca_1, addlabels=TRUE, barfill="steelblue", barcolor="darkblue")
```

```
# Create loadings plot
library(ggplot2)
loadings <- pca_1$rotation[, c("PC1", "PC2")]
ggplot(data=loadings, aes(x=PC1, y=PC2, label=rownames(loadings)))
  geom_point(size=3, color="red")
  geom_text(vjust=1.5, size=3)
  geom_vline(xintercept=0, lty="dashed")
  geom_hline(yintercept=0, lty="dashed")
  ggtitle("Loadings Plot")
  theme_bg()
```

```
# Create biplot with PC1 and PC2
library(ggfactoextra)
fviz_pca_biplot(pca_1, geom.ind="point", pointsize=2, palette=c("red", "blue", "green", "purple"), addEllipses=TRUE, ellipse.level=0.95, legend.title="Cluster")
```

```
# Create scree plot
library(factoextra)
fviz_eig(pca_1, addlabels=TRUE, barfill="steelblue", barcolor="darkblue")
```

Dim1
Dim2
Dim3
Dim4
Dim5
Dim6
Dim7
Dim8
Dim9
Dim10
Dim11
Dim12
Dim13
Dim14
Dim15
Dim16
Dim17
Dim18
Dim19
Dim20
Dim21
Dim22
Dim23
Dim24
Dim25
Dim26
Dim27
Dim28
Dim29
Dim30
Dim31
Dim32
Dim33
Dim34
Dim35
Dim36
Dim37
Dim38
Dim39
Dim40
Dim41
Dim42
Dim43
Dim44
Dim45
Dim46
Dim47
Dim48
Dim49
Dim50
Dim51
Dim52
Dim53
Dim54
Dim55
Dim56
Dim57
Dim58
Dim59
Dim60
Dim61
Dim62
Dim63
Dim64
Dim65
Dim66
Dim67
Dim68
Dim69
Dim70
Dim71
Dim72
Dim73
Dim74
Dim75
Dim76
Dim77
Dim78
Dim79
Dim80
Dim81
Dim82
Dim83
Dim84
Dim85
Dim86
Dim87
Dim88
Dim89
Dim90
Dim91
Dim92
Dim93
Dim94
Dim95
Dim96
Dim97
Dim98
Dim99
Dim100
Dim101
Dim102
Dim103
Dim104
Dim105
Dim106
Dim107
Dim108
Dim109
Dim110
Dim111
Dim112
Dim113
Dim114
Dim115
Dim116
Dim117
Dim118
Dim119
Dim120
Dim121
Dim122
Dim123
Dim124
Dim125
Dim126
Dim127
Dim128
Dim129
Dim130
Dim131
Dim132
Dim133
Dim134
Dim135
Dim136
Dim137
Dim138
Dim139
Dim140
Dim141
Dim142
Dim143
Dim144
Dim145
Dim146
Dim147
Dim148
Dim149
Dim150
Dim151
Dim152
Dim153
Dim154
Dim155
Dim156
Dim157
Dim158
Dim159
Dim160
Dim161
Dim162
Dim163
Dim164
Dim165
Dim166
Dim167
Dim168
Dim169
Dim170
Dim171
Dim172
Dim173
Dim174
Dim175
Dim176
Dim177
Dim178
Dim179
Dim180
Dim181
Dim182
Dim183
Dim184
Dim185
Dim186
Dim187
Dim188
Dim189
Dim190
Dim191
Dim192
Dim193
Dim194
Dim195
Dim196
Dim197
Dim198
Dim199
Dim200
Dim201
Dim202
Dim203
Dim204
Dim205
Dim206
Dim207
Dim208
Dim209
Dim210
Dim211
Dim212
Dim213
Dim214
Dim215
Dim216
Dim217
Dim218
Dim219
Dim220
Dim221
Dim222
Dim223
Dim224
Dim225
Dim226
Dim227
Dim228
Dim229
Dim230
Dim231
Dim232
Dim233
Dim234
Dim235
Dim236
Dim237
Dim238
Dim239
Dim240
Dim241
Dim242
Dim243
Dim244
Dim245
Dim246
Dim247
Dim248
Dim249
Dim250
Dim251
Dim252
Dim253
Dim254
Dim255
Dim256
Dim257
Dim258
Dim259
Dim260
Dim261
Dim262
Dim263
Dim264
Dim265
Dim266
Dim267
Dim268
Dim269
Dim270
Dim271
Dim272
Dim273
Dim274
Dim275
Dim276
Dim277
Dim278
Dim279
Dim280
Dim281
Dim282
Dim283
Dim284
Dim285
Dim286
Dim287
Dim288
Dim289
Dim290
Dim291
Dim292
Dim293
Dim294
Dim295
Dim296
Dim297
Dim298
Dim299
Dim300
Dim301
Dim302
Dim303
Dim304
Dim305
Dim306
Dim307
Dim308
Dim309
Dim310
Dim311
Dim312
Dim313
Dim314
Dim315
Dim316
Dim317
Dim318
Dim319
Dim320
Dim321
Dim322
Dim323
Dim324
Dim325
Dim326
Dim327
Dim328
Dim329
Dim330
Dim331
Dim332
Dim333
Dim334
Dim335
Dim336
Dim337
Dim338
Dim339
Dim340
Dim341
Dim342
Dim343
Dim344
Dim345
Dim346
Dim347
Dim348
Dim349
Dim350
Dim351
Dim352
Dim353
Dim354
Dim355
Dim356
Dim357
Dim358
Dim359
Dim360
Dim361
Dim362
Dim363
Dim364
Dim365
Dim366
Dim367
Dim368
Dim369
Dim370
Dim371
Dim372
Dim373
Dim374
Dim375
Dim376
Dim377
Dim378
Dim379
Dim380
Dim381
Dim382
Dim383
Dim384
Dim385
Dim386
Dim387
Dim388
Dim389
Dim390
Dim391
Dim392
Dim393
Dim394
Dim395
Dim396
Dim397
Dim398
Dim399
Dim400
Dim401
Dim402
Dim403
Dim404
Dim405
Dim406
Dim407
Dim408
Dim409
Dim410
Dim411
Dim412
Dim413
Dim414
Dim415
Dim416
Dim417
Dim418
Dim419
Dim420
Dim421
Dim422
Dim423
Dim424
Dim425
Dim426
Dim427
Dim428
Dim429
Dim430
Dim431
Dim432
Dim433
Dim434
Dim435
Dim436
Dim437
Dim438
Dim439
Dim440
Dim441
Dim442
Dim443
Dim444
Dim445
Dim446
Dim447
Dim448
Dim449
Dim450
Dim451
Dim452
Dim453
Dim454
Dim455
Dim456
Dim457
Dim458
Dim459
Dim460
Dim461
Dim462
Dim463
Dim464
Dim465
Dim466
Dim467
Dim468
Dim469
Dim470
Dim471
Dim472
Dim473
Dim474
Dim475
Dim476
Dim477
Dim478
Dim479
Dim480
Dim481
Dim482
Dim483
Dim484
Dim485
Dim486
Dim487
Dim488
Dim489
Dim490
Dim491
Dim492
Dim493
Dim494
Dim495
Dim496
Dim497
Dim498
Dim499
Dim500
Dim501
Dim502
Dim503
Dim504
Dim505
Dim506
Dim507
Dim508
Dim509
Dim510
Dim511
Dim512
Dim513
Dim514
Dim515
Dim516
Dim517
Dim518
Dim519
Dim520
Dim521
Dim522
Dim523
Dim524
Dim525
Dim526
Dim527
Dim528
Dim529
Dim530
Dim531
Dim532
Dim533
Dim534
Dim535
Dim536
Dim537
Dim538
Dim539
Dim540
Dim541
Dim542
Dim543
Dim544
Dim545
Dim546
Dim547
Dim548
Dim549
Dim550
Dim551
Dim552
Dim553
Dim554
Dim555
Dim556
Dim557
Dim558
Dim559
Dim560
Dim561
Dim562
Dim563
Dim564
Dim565
Dim566
Dim567
Dim568
Dim569
Dim570
Dim571
Dim572
Dim573
Dim574
Dim575
Dim576
Dim577
Dim578
Dim579
Dim580
Dim581
Dim582
Dim583
Dim584
Dim585
Dim586
Dim587
Dim588
Dim589
Dim590
Dim591
Dim592
Dim593
Dim594
Dim595
Dim596
Dim597
Dim598
Dim599
Dim600
Dim601
Dim602
Dim603
Dim604
Dim605
Dim606
Dim607
Dim608
Dim609
Dim610
Dim611
Dim612
Dim613
Dim614
Dim615
Dim616
Dim617
Dim618
Dim619
Dim620
Dim621
Dim622
Dim623
Dim624
Dim625
Dim626
Dim627
Dim628
Dim629
Dim630
Dim631
Dim632
Dim633
Dim634
Dim635
Dim636
Dim637
Dim638
Dim639
Dim640
Dim641
Dim642
Dim643
Dim644
Dim645
Dim646
Dim647
Dim648
Dim649
Dim650
Dim651
Dim652
Dim653
Dim654
Dim655
Dim656
Dim657
Dim658
Dim659
Dim660
Dim661
Dim662
Dim663
Dim664
Dim665
Dim666
Dim667
Dim668
Dim669
Dim670
Dim671
Dim672
Dim673
Dim674
Dim675
Dim676
Dim677
Dim678
Dim679
Dim680
Dim681
Dim682
Dim683
Dim684
Dim685
Dim686
Dim687
Dim688
Dim689
Dim690
Dim691
Dim692
Dim693
Dim694
Dim695
Dim696
Dim697
Dim698
Dim699
Dim700
Dim701
Dim702
Dim703
Dim704
Dim705
Dim706
Dim707
Dim708
Dim709
Dim710
Dim711
Dim712
Dim713
Dim714
Dim715
Dim716
Dim717
Dim718
Dim719
Dim720
Dim721
Dim722
Dim723
Dim724
Dim725
Dim726
Dim727
Dim728
Dim729
Dim730
Dim731
Dim732
Dim733
Dim734
Dim735
Dim736
Dim737
Dim738
Dim739
Dim740
Dim741
Dim742
Dim743
Dim744
Dim745
Dim746
Dim747
Dim748
Dim749
Dim750
Dim751
D

```
# # Correlation Variables and Plot
# # Correlation Variables and Plot
# # Correlation Variables and Plot
```

```
# Standardize features
numeric_features = ['Income', 'Frequency', 'Monetary', 'Tenure']
scaler = StandardScaler()
df_scaled[numeric_features] = scaler.fit_transform(df[numeric_features])

# Calculate optimal number of clusters using elbow method
inertia = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(df_scaled)
    inertia.append(kmeans.inertia_)

# Plot elbow curve
plt.figure(figure=(10,6))
plt.plot(range(1, 11), inertia, 'bo-')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.title('Elbow Method for Optimal Clusters')
plt.grid(True)
plt.show()
```

```
# Apply K-means with optimal number of clusters (4 based on the elbow method)
optimal_k = 4
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
df['Cluster'] = kmeans.fit_predict(df_scaled)

# Calculate cluster centers
cluster_centers = scaler.inverse_transform(kmeans.cluster_centers_)

# Create DataFrame with cluster centers
cluster_df = pd.DataFrame(cluster_centers, columns=numeric_features)
cluster_df['Cluster'] = range(optimal_k)

# Calculate cluster sizes
cluster_sizes = df['Cluster'].value_counts()
cluster_df['Size'] = cluster_sizes.sort_index.values

print("Cluster Centers:")
print(cluster_df.round(2))
```

```
Cluster Centers:
   Income  Frequency  Monetary  Tenure  Cluster  Size
0   14.00       1.00      0.00    0.00        0   ...
1   14.00       2.00      2.00    2.00        1   ...
2   14.00       3.00      3.00    3.00        2   ...
3   14.00       4.00      4.00    4.00        3   ...
```

```
# Create visualization of cluster centers
plt.figure(figure=(12,6))
sns.heatmap(cluster_df[numeric_features].T, annot=True, cmap='YlGnBu')
plt.title('Cluster Characteristics')
plt.xlabel('Cluster')
plt.ylabel('Features')
plt.show()
```

```
# Create pairplot to visualize relationships between features
sns.pairplot(df[numeric_features + ['Cluster']], hue='Cluster', palette='viridis')
plt.suptitle('Pairwise Relationships by Cluster', y=1.02)
plt.show
```

```
# Calculate and display cluster statistics
cluster_stats = df.groupby('Cluster')[numeric_features].agg(['mean', 'std', 'min', 'max'])
print("Detailed Cluster Statistics:")
print(cluster_stats.round(2))
```

```
Detailed Cluster Statistics:
         Income                   Frequency                   Monetary                   Tenure
           mean  std  min  max      mean  std  min  max      mean  std  min  max      mean  std  min  max
Cluster
0           ...  ...  ...  ...       ...  ...  ...  ...       ...  ...  ...  ...       ...  ...  ...  ...
1           ...  ...  ...  ...       ...  ...  ...  ...       ...  ...  ...  ...       ...  ...  ...  ...
2           ...  ...  ...  ...       ...  ...  ...  ...       ...  ...  ...  ...       ...  ...  ...  ...
3           ...  ...  ...  ...       ...  ...  ...  ...       ...  ...  ...  ...       ...  ...  ...  ...
```

```
# Save results to CSV
df.to_csv('customer_segments.csv')

# Save cluster centers
cluster_df.to_csv('cluster_centers.csv')

# Save cluster statistics
cluster_stats.to_csv('cluster_statistics.csv')
```

```
# Create a function to predict cluster for new customers
def predict_cluster(income, frequency, monetary, tenure):
    new_customer = pd.DataFrame([[income, frequency, monetary, tenure]],
                                columns=numeric_features)
    new_customer_scaled = scaler.transform(new_customer)
    cluster = kmeans.predict(new_customer_scaled)[0]
    return cluster

# Example usage
print("Example: Predicting cluster for new customer")
print("Cluster:", predict_cluster(14, 1, 0, 0))
```

```
Example: Predicting cluster for new customer
Cluster: 0
```

```
# Calculate correlation between features
correlation = df[numeric_features].corr()
print("Correlation between features:")
print(correlation.round(2))
```

```
Correlation between features:
        Income  Frequency  Monetary  Tenure
Income    1.00       0.00      0.00    0.00
Frequency 0.00       1.00      0.00    0.00
Monetary  0.00       0.00      1.00    0.00
Tenure    0.00       0.00      0.00    1.00
```

```
# Create pairplot with correlation coefficients
g = sns.pairplot(df[numeric_features + ['Cluster']], hue='Cluster', palette='viridis')
g.fig.suptitle('Pairwise Relationships by Cluster', y=1.02)
plt.show()
```



```
# Calculate silhouette score
from sklearn.metrics import silhouette_score
silhouette_avg = silhouette_score(df_scaled, df['Cluster'])
print("Silhouette Score:", silhouette_avg)
```

```
Silhouette Score: 0.[illegible]
```

```
# Save the complete analysis
df.to_csv('customer_segments_with_clusters.csv')
cluster_df.to_csv('cluster_centers_with_sizes.csv')
cluster_stats.to_csv('cluster_statistics_detailed.csv')
```

```
# Create a summary report
print("Customer Segmentation Analysis Summary")
print("=====================================")
print("Number of customers:", len(df))
print("Number of clusters:", optimal_k)
print("Silhouette Score:", silhouette_avg)
print("\nCluster Sizes:")
print(cluster_sizes.sort_index())
```

```
Customer Segmentation Analysis Summary
=====================================
Number of customers: [illegible]
Number of clusters: 4
Silhouette Score: [illegible]

Cluster Sizes:
[illegible]
```

```
# Create a function to visualize new customer in relation to clusters
def visualize_new_customer(income, frequency, monetary, tenure):
    new_customer = pd.DataFrame([[income, frequency, monetary, tenure]],
                                columns=numeric_features)
    new_customer_scaled = scaler.transform(new_customer)
    cluster = kmeans.predict(new_customer_scaled)[0]

    plt.figure(figure=(12,6))
    sns.pairplot(df[numeric_features + ['Cluster']], hue='Cluster', palette='viridis')
    plt.suptitle('Pairwise Relationships by Cluster', y=1.02)
    plt.show

    return cluster

# Example usage
print("Example: Visualizing new customer")
cluster = visualize_new_customer(14, 1, 0, 0)
print("Cluster:", cluster)
```

```
Example: Visualizing new customer
Cluster: 0
```



```
# Create a function to compare new customer with cluster averages
def compare_with_cluster(income, frequency, monetary, tenure):
    new_customer = pd.DataFrame([[income, frequency, monetary, tenure]],
                                columns=numeric_features)
    new_customer_scaled = scaler.transform(new_customer)
    cluster = kmeans.predict(new_customer_scaled)[0]

    print("Comparison with cluster", cluster, "averages:")
    for feature in numeric_features:
        cluster_mean = cluster_df[cluster][feature]
        new_value = new_customer[feature]
        diff = new_value - cluster_mean
        print(f"{feature}: {new_value} vs {cluster_mean:.2f} (difference: {diff:.2f})")

# Example usage
print("Example: Comparing new customer with cluster averages")
compare_with_cluster(14, 1, 0, 0)
```

```
Example: Comparing new customer with cluster averages
Comparison with cluster 0 averages:
Income: 14 vs [illegible] (difference: [illegible])
Frequency: 1 vs [illegible] (difference: [illegible])
Monetary: 0 vs [illegible] (difference: [illegible])
Tenure: 0 vs [illegible] (difference: [illegible])
```

```
# Save the complete analysis with all components
df.to_csv('customer_segments_with_clusters.csv')
cluster_df.to_csv('cluster_centers_with_sizes.csv')
cluster_stats.to_csv('cluster_statistics_detailed.csv')
correlation.to_csv('correlation_matrix.csv')
```

```
# Create a final summary report
print("Customer Segmentation Analysis Summary")
print("=====================================")
print("Number of customers:", len(df))
print("Number of clusters:", optimal_k)
print("Silhouette Score:", silhouette_avg)
print("\nCluster Sizes:")
print(cluster_sizes.sort_index())
print("\nCorrelation between features:")
print(correlation.round(2))
```

```
Customer Segmentation Analysis Summary
=====================================
Number of customers: [illegible]
Number of clusters: 4
Silhouette Score: [illegible]

Cluster Sizes:
[illegible]

Correlation between features:
[illegible]
```

```
# Create a function to predict and visualize new customer
def analyze_new_customer(income, frequency, monetary, tenure):
    new_customer = pd.DataFrame([[income, frequency, monetary, tenure]],
                                columns=numeric_features)
    new_customer_scaled = scaler.transform(new_customer)
    cluster = kmeans.predict(new_customer_scaled)[0]

    print("Analysis for new customer:")
    print("Predicted cluster:", cluster)
    print("\nComparison with cluster", cluster, "averages:")
    for feature in numeric_features:
        cluster_mean = cluster_df[cluster][feature]
        new_value = new_customer[feature]
        diff = new_value - cluster_mean
        print(f"{feature}: {new_value} vs {cluster_mean:.2f} (difference: {diff:.2f})")

    plt.figure(figure=(12,6))
    sns.pairplot(df[numeric_features + ['Cluster']], hue='Cluster', palette='viridis')
    plt.suptitle('Pairwise Relationships by Cluster', y=1.02)
    plt.show

    return cluster

# Example usage
print("Example: Analyzing new customer")
analyze_new_customer(14, 1, 0, 0)
```

```
Example: Analyzing new customer
Analysis for new customer:
Predicted cluster: 0

Comparison with cluster 0 averages:
Income: 14 vs [illegible] (difference: [illegible])
Frequency: 1 vs [illegible] (difference: [illegible])
Monetary: 0 vs [illegible] (difference: [illegible])
Tenure: 0 vs [illegible] (difference: [illegible])
```



```
# Save the analysis functions
import joblib
joblib.dump(predict_cluster, 'predict_cluster.pkl')
joblib.dump(visualize_new_customer, 'visualize_new_customer.pkl')
joblib.dump(compare_with_cluster, 'compare_with_cluster.pkl')
joblib.dump(analyze_new_customer, 'analyze_new_customer.pkl')
```

```
['predict_cluster.pkl']
['visualize_new_customer.pkl']
['compare_with_cluster.pkl']
['analyze_new_customer.pkl']
```

```
# Create a final summary report
print("Customer Segmentation Analysis Summary")
print("=====================================")
print("Number of customers:", len(df))
print("Number of clusters:", optimal_k)
print("Silhouette Score:", silhouette_avg)
print("\nCluster Sizes:")
print(cluster_sizes.sort_index())
print("\nCorrelation between features:")
print(correlation.round(2))
```

```
Customer Segmentation Analysis Summary
=====================================
Number of customers: [illegible]
Number of clusters: 4
Silhouette Score: [illegible]

Cluster Sizes:
[illegible]

Correlation between features:
[illegible]
```

```
# Save the complete analysis
df.to_csv('customer_segments_with_clusters.csv')
cluster_df.to_csv('cluster_centers_with_sizes.csv')
cluster_stats.to_csv('cluster_statistics_detailed.csv')
correlation.to_csv('correlation_matrix.csv')
```

```
# Create a final summary report
print("Customer Segmentation Analysis Summary")
print("=====================================")
print("Number of customers:", len(df))
print("Number of clusters:", optimal_k)
print("Silhouette Score:", silhouette_avg)
print("\nCluster Sizes:")
print(cluster_sizes.sort_index())
print("\nCorrelation between features:")
print(correlation.round(2))
```

```
Customer Segmentation Analysis Summary
=====================================
Number of customers: [illegible]
Number of clusters: 4
Silhouette Score: [illegible]

Cluster Sizes:
[illegible]

Correlation between features:
[illegible]
```

```
# Save the analysis functions
import joblib
joblib.dump(predict_cluster, 'predict_cluster.pkl')
joblib.dump(visualize_new_customer, 'visualize_new_customer.pkl')
joblib.dump(compare_with_cluster, 'compare_with_cluster.pkl')
joblib.dump(analyze_new_customer, 'analyze_new_customer.pkl')
```

```
['predict_cluster.pkl']
['visualize_new_customer.pkl']
['compare_with_cluster.pkl']
['analyze_new_customer.pkl']
```

```
# Create a final summary report
print("Customer Segmentation Analysis Summary")
print("=====================================")
print("Number of customers:", len(df))
print("Number of clusters:", optimal_k)
print("Silhouette Score:", silhouette_avg)
print("\nCluster Sizes:")
print(cluster_sizes.sort_index())
print("\nCorrelation between features:")
print(correlation.round(2))
```

```
Customer Segmentation Analysis Summary
=====================================
Number of customers: [illegible]
Number of clusters: 4
Silhouette Score: [illegible]

Cluster Sizes:
[illegible]

Correlation between features:
[illegible]
```

```
# Save the complete analysis
df.to_csv('customer_segments_with_clusters.csv')
cluster_df.to_csv('cluster_centers_with_sizes.csv')
cluster_stats.to_csv('cluster_statistics_detailed.csv')
correlation.to_csv('correlation_matrix.csv')
```

```
# Create a final summary report
print("Customer Segmentation Analysis Summary")
print("=====================================")
print("Number of customers:", len(df))
print("Number of clusters:", optimal_k)
print("Silhouette Score:", silhouette_avg)
print("\nCluster Sizes:")
print(cluster_sizes.sort_index())
print("\nCorrelation between features:")
print(correlation.round(2))
```

```
Customer Segmentation Analysis Summary
=====================================
Number of customers: [illegible]
Number of clusters: 4
Silhouette Score: [illegible]

Cluster Sizes:
[illegible]

Correlation between features:
[illegible]
```

```
# Save the analysis functions
import joblib
joblib.dump(predict_cluster, 'predict_cluster.pkl')
joblib.dump(visualize_new_customer, 'visualize_new_customer.pkl')
joblib.dump(compare_with_cluster, 'compare_with_cluster.pkl')
joblib.dump(analyze_new_customer, 'analyze_new_customer.pkl')
```

```
['predict_cluster.pkl']
['visualize_new_customer.pkl']
['compare_with_cluster.pkl']
['analyze_new_customer.pkl']
```

```
# Create a final summary report
print("Customer Segmentation Analysis Summary")
print("=====================================")
print("Number of customers:", len(df))
print("Number of clusters:", optimal_k)
print("Silhouette Score:", silhouette_avg)
print("\nCluster Sizes:")
print(cluster_sizes.sort_index())
print("\nCorrelation between features:")
print(correlation.round(2))
```

```
Customer Segmentation Analysis Summary
=====================================
Number of customers: [illegible]
Number of clusters: 4
Silhouette Score: [illegible]

Cluster Sizes:
[illegible]

Correlation between features:
[illegible]
```

```
# Save the complete analysis
df.to_csv('customer_segments_with_clusters.csv')
cluster_df.to_csv('cluster_centers_with_sizes.csv')
cluster_stats.to_csv('cluster_statistics_detailed.csv')
correlation.to_csv('correlation_matrix.csv')
```

```
# Create a final summary report
print("Customer Segmentation Analysis Summary")
print("=====================================")
print("Number of customers:", len(df))
print("Number of clusters:", optimal_k)
print("Silhouette Score:", silhouette_avg)
print("\nCluster Sizes:")
print(cluster_sizes.sort_index())
print("\nCorrelation between features:")
print(correlation.round(2))
```

```
Customer Segmentation Analysis Summary
=====================================
Number of customers: [illegible]
Number of clusters: 4
Silhouette Score: [illegible]

Cluster Sizes:
[illegible]

Correlation between features:
[illegible]
```

```
# Save the analysis functions
import joblib
joblib.dump(predict_cluster, 'predict_cluster.pkl')
joblib.dump(visualize_new_customer, 'visualize_new_customer.pkl')
joblib.dump(compare_with_cluster, 'compare_with_cluster.pkl')
joblib.dump(analyze_new_customer, 'analyze_new_customer.pkl')
```

```
['predict_cluster.pkl']
['visualize_new_customer.pkl']
['compare_with_cluster.pkl']
['analyze_new_customer.pkl']
```

```
# Create a final summary report
print("Customer Segmentation Analysis Summary")
print("=====================================")
print("Number of customers:", len(df))
print("Number of clusters:", optimal_k)
print("Silhouette Score:", silhouette_avg)
print("\nCluster Sizes:")
print(cluster_sizes.sort_index())
print("\nCorrelation between features:")
print(correlation.round(2))
```

```
Customer Segmentation Analysis Summary
=====================================
Number of customers: [illegible]
Number of clusters: 4
Silhouette Score: [illegible]

Cluster Sizes:
[illegible]

Correlation between features:
[illegible]
```

```
# Save the complete analysis
df.to_csv('customer_segments_with_clusters.csv')
cluster_df.to_csv('cluster_centers_with_sizes.csv')
cluster_stats.to_csv('cluster_statistics_detailed.csv')
correlation.to_csv('correlation_matrix.csv')
```

```
# Create a final summary report
print("Customer Segmentation Analysis Summary")
print("=====================================")
print("Number of customers:", len(df))
print("Number of clusters:", optimal_k)
print("Silhouette Score:", silhouette_avg)
print("\nCluster Sizes:")
print(cluster_sizes.sort_index())
print("\nCorrelation between features:")
print(correlation.round(2))
```

```
Customer Segmentation Analysis Summary
=====================================
Number of customers: [illegible]
Number of clusters: 4
Silhouette Score: [illegible]

Cluster Sizes:
[illegible]

Correlation between features:
[illegible]
```

```
# Save the analysis functions
import joblib
joblib.dump(predict_cluster, 'predict_cluster.pkl')
joblib.dump(visualize_new_customer, 'visualize_new_customer.pkl')
joblib.dump(compare_with_cluster, 'compare_with_cluster.pkl')
joblib.dump(analyze_new_customer, 'analyze_new_customer.pkl')
```

```
['predict_cluster.pkl']
['visualize_new_customer.pkl']
['compare_with_cluster.pkl']
['analyze_new_customer.pkl']
```

SOM

```
In [1]: %%time
        %matplotlib inline
        import numpy as np, matplotlib, os
        os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
        os.environ['OMP_NUM_THREADS'] = '4'
        os.environ['KERAS_BACKEND'] = 'tensorflow'
        os.environ['THEANO_FLAGS'] = 'device=cpu'
        import matplotlib.pyplot as plt
        plt.rcParams['figure.dpi'] = 150
        plt.rcParams['savefig.dpi'] = 150
        plt.rcParams['image.interpolation'] = 'none'
        plt.rcParams['image.cmap'] = 'gray'
        plt.rcParams['axes.grid'] = False
        plt.rcParams['axes.xmargin'] = 0
        plt.rcParams['axes.ymargin'] = 0
        plt.rcParams['ytick.major.size'] = 0
        plt.rcParams['xtick.major.size'] = 0
        plt.rcParams['axes.titlesize'] = 9
        plt.rcParams['figure.titlesize'] = 9
        plt.rcParams['axes.titlepad'] = 4
        plt.rcParams['figure.subplot.wspace'] = 0.05
        plt.rcParams['figure.subplot.hspace'] = 0.05
        plt.rcParams['figure.subplot.bottom'] = 0.05
        plt.rcParams['figure.subplot.top'] = 0.95
        plt.rcParams['figure.subplot.left'] = 0.05
        plt.rcParams['figure.subplot.right'] = 0.95
        plt.rcParams['figure.figsize'] = 3.0000, 3.0000
        plt.rcParams['savefig.bbox'] = 'tight'
        plt.rcParams['savefig.pad_inches'] = 0.0000
        plt.rcParams['savefig.jpeg_quality'] = 100
        plt.rcParams['savefig.jpeg_progressive'] = True
        plt.rcParams['savefig.tiff_compression'] = 'tiffpack'
        plt.rcParams['savefig.tiff_compression_level'] = 9
        plt.rcParams['savefig.tiff_tile_size'] = 256
        plt.rcParams['savefig.tiff_rows_per_strip'] = 256
        plt.rcParams['savefig.webp_quality'] = 100
        plt.rcParams['savefig.webp_method'] = 6
        plt.rcParams['savefig.webp_lossless'] = True
        plt.rcParams['savefig.png_compression'] = 9
        plt.rcParams['savefig.png_filter'] = 'paeth'
        plt.rcParams['savefig.png_strategy'] = 'rle'
        plt.rcParams['savefig.png_dpi'] = 150
        plt.rcParams['savefig.png_text'] = False
        plt.rcParams['savefig.png_optimize'] = True
        plt.rcParams['savefig.png_interlace'] = False
        plt.rcParams['savefig.png_compression_level'] = 9
        plt.rcParams['savefig.png_compression_strategy'] = 'rle'
        plt.rcParams['savefig.png_compression_filter'] = 'paeth'
        plt.rcParams['savefig.png_compression_dpi'] = 150
        plt.rcParams['savefig.png_compression_text'] = False
        plt.rcParams['savefig.png_compression_optimize'] = True
        plt.rcParams['savefig.png_compression_interlace'] = False
        plt.rcParams['savefig.png_compression_compression_level'] = 9
        plt.rcParams['savefig.png_compression_compression_strategy'] = 'rle'
        plt.rcParams['savefig.png_compression_compression_filter'] = 'paeth'
        plt.rcParams['savefig.png_compression_compression_dpi'] = 150
        plt.rcParams['savefig.png_compression_compression_text'] = False
        plt.rcParams['savefig.png_compression_compression_optimize'] = True
        plt.rcParams['savefig.png_compression_compression_interlace'] = False
        plt.rcParams['savefig.png_compression_compression_compression_level'] = 9
        plt.rcParams['savefig.png_compression_compression_compression_strategy'] = 'rle'
        plt.rcParams['savefig.png_compression_compression_compression_filter'] = 'paeth'
        plt.rcParams['savefig.png_compression_compression_compression_dpi'] = 150
        plt.rcParams['savefig.png_compression_compression_compression_text'] = False
        plt.rcParams['savefig.png_compression_compression_compression_optimize'] = True
        plt.rcParams['savefig.png_compression_compression_compression_interlace'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_level'] = 9
        plt.rcParams['savefig.png_compression_compression_compression_compression_strategy'] = 'rle'
        plt.rcParams['savefig.png_compression_compression_compression_compression_filter'] = 'paeth'
        plt.rcParams['savefig.png_compression_compression_compression_compression_dpi'] = 150
        plt.rcParams['savefig.png_compression_compression_compression_compression_text'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_optimize'] = True
        plt.rcParams['savefig.png_compression_compression_compression_compression_interlace'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_level'] = 9
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_strategy'] = 'rle'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_filter'] = 'paeth'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_dpi'] = 150
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_text'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_optimize'] = True
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_interlace'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_level'] = 9
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_strategy'] = 'rle'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_filter'] = 'paeth'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_dpi'] = 150
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_text'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_optimize'] = True
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_interlace'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_level'] = 9
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_strategy'] = 'rle'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_filter'] = 'paeth'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_dpi'] = 150
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_text'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_optimize'] = True
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_interlace'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_level'] = 9
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_strategy'] = 'rle'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_filter'] = 'paeth'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_dpi'] = 150
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_text'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_optimize'] = True
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_interlace'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_level'] = 9
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_strategy'] = 'rle'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_filter'] = 'paeth'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_dpi'] = 150
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_text'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_optimize'] = True
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_interlace'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_level'] = 9
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_strategy'] = 'rle'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_filter'] = 'paeth'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_dpi'] = 150
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_text'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_optimize'] = True
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_interlace'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_level'] = 9
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_strategy'] = 'rle'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_filter'] = 'paeth'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_dpi'] = 150
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_text'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_optimize'] = True
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_interlace'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_level'] = 9
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_strategy'] = 'rle'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_filter'] = 'paeth'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_dpi'] = 150
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_text'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_optimize'] = True
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_interlace'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_level'] = 9
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_strategy'] = 'rle'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_filter'] = 'paeth'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_dpi'] = 150
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_text'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_optimize'] = True
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_interlace'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_level'] = 9
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_strategy'] = 'rle'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_filter'] = 'paeth'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_dpi'] = 150
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_text'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_optimize'] = True
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_interlace'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_level'] = 9
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_strategy'] = 'rle'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_filter'] = 'paeth'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_dpi'] = 150
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_text'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_optimize'] = True
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_interlace'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_level'] = 9
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_strategy'] = 'rle'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_filter'] = 'paeth'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_dpi'] = 150
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_text'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_optimize'] = True
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_interlace'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_level'] = 9
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_strategy'] = 'rle'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_filter'] = 'paeth'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_dpi'] = 150
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_text'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_optimize'] = True
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_interlace'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_level'] = 9
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_strategy'] = 'rle'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_filter'] = 'paeth'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_dpi'] = 150
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_text'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_optimize'] = True
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_interlace'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_level'] = 9
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_strategy'] = 'rle'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_filter'] = 'paeth'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_dpi'] = 150
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_text'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_optimize'] = True
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_interlace'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_level'] = 9
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_strategy'] = 'rle'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_filter'] = 'paeth'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_dpi'] = 150
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_text'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_optimize'] = True
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_interlace'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_level'] = 9
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_strategy'] = 'rle'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_filter'] = 'paeth'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_dpi'] = 150
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_text'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_optimize'] = True
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_interlace'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_level'] = 9
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_strategy'] = 'rle'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_filter'] = 'paeth'
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_dpi'] = 150
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_text'] = False
        plt.rcParams['savefig.png_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_compression_optimize'] = True
        plt.rcParams['savefig.png_compression
```

In [1]:
```
# Hierarchical clustering
# Ward, Average, Single, Complete, Centroid, Median
# ward.D ward.D2 single complete average mcquitty median centroid
rm(list=ls())
for(p in c("dendextend","ape")){cat("Trying to load",p,"\n")
  if(!require(p,quietly=TRUE,character.only=TRUE)){cat("Failed to load",p,"\n")
    if(!exists("ask.update.packages")){ask.update.packages=TRUE}
    if(ask.update.packages){update.packages(ask=TRUE,checkBuilt=TRUE,repos=c("http://cran.rstudio.com"))
    ask.update.packages=FALSE}
    if(!require(p,quietly=TRUE,character.vs.fortune=TRUE)){stop("Failed to load",p,"\n")}
  }else{cat("Loaded",p,"\n")}
}
gc(full=TRUE)
```

In [2]:
```
dist=read.csv("distance.csv",header=TRUE,row.names=1,check.names=FALSE)
dist=as.dist(dist)
```

In [3]:
```
for(m in c("ward.D","ward.D2","single","complete","average","mcquitty","median","centroid")){
  cat("Method:",m,"\n")
  hc=hclust(dist,method=m)
  cat("Cophenetic correlation:",cor(cophenetic(hc),dist),"\n")
  plot(hc,cex=0.6,hang=-1,main=paste0("Method: ",m))
  rect.hclust(hc,k=2,border=c("red","blue"))
}
```

```
Method: ward.D
Cophenetic correlation: 0.8966666
Method: ward.D2
Cophenetic correlation: 0.9666667
Method: single
Cophenetic correlation: 0.9666667
Method: complete
Cophenetic correlation: 0.9666667
Method: average
Cophenetic correlation: 0.9666667
Method: mcquitty
Cophenetic correlation: 0.9666667
Method: median
Cophenetic correlation: 0.9666667
Method: centroid
Cophenetic correlation: 0.9666667
```

In [4]:
```
# Hierarchical clustering with ward.D2 method
hc=hclust(dist,method="ward.D2")
cut=cutree(hc,k=2)
table(cut)
```

```
cut
 1 2
10 5
```

In [5]:
```
# Plot dendrogram with 2 clusters
plot(hc,cex=0.6,hang=-1,main="Hierarchical Clustering with Ward.D2 Method")
rect.hclust(hc,k=2,border=c("red","blue"))
```

In [6]:
```
# Hierarchical Clustering with Ward.D2 method
hc=hclust(dist,method="ward.D2")
cut=cutree(hc,k=2)
table(cut)
plot(hc,cex=0.6,hang=-1,main="Hierarchical Clustering with Ward.D2 Method")
rect.hclust(hc,k=2,border=c("red","blue"))
```

```
cut
 1 2
10 5
```

In [7]:
```
# Hierarchical DBSCAN (HDBSCAN)
rm(list=ls())
library(dbscan)
# Load data
dist=read.csv("distance.csv",header=TRUE,row.names=1,check.names=FALSE)
dist=as.dist(dist)
# Run HDBSCAN with minimum cluster size = 2
set.seed(123)
hdb=hdbscan(dist,minPts=2)
cat("Cluster assignments:\n")
table(hdb$cluster)
cat("Outlier scores:\n")
hdb$outlier.scores
```

```
Cluster assignments:
 1 2
10 5
Outlier scores:
 [1] 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
 [9] 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
```

In [8]:
```
# Plot dendrogram
plot(hdb,gap=0.5)
```

In [9]:
```
# Extract cluster assignments
cut=hdb$cluster
# Create dendrogram with colored branches
library(dendextend)
dend=as.dendrogram(hdb)
dend=color_branches(dend,k=length(unique(cut[cut!=0])),col=c("red","blue"))
# Plot dendrogram
plot(dend,main="HDBSCAN Dendrogram")
```

In [10]:
```
# Load data
rm(list=ls())
dist=read.csv("distance.csv",header=TRUE,row.names=1,check.names=FALSE)
dist=as.dist(dist)
# Run PAM clustering
library(cluster)
set.seed(123)
pam=pam(dist,k=2)
cat("Cluster assignments:\n")
table(pam$clustering)
cat("Silhouette width:",pam$silinfo$avg.width,"\n")
```

```
Cluster assignments:
 1 2
10 5
Silhouette width: 0.7333333
```

In [11]:
```
# Plot silhouette
plot(silhouette(pam),main="PAM Silhouette Plot")
```


In [12]:
```
# Load data
rm(list=ls())
dist=read.csv("distance.csv",header=TRUE,row.names=1,check.names=FALSE)
dist=as.dist(dist)
# Run PAM clustering
library(cluster)
set.seed(123)
pam=pam(dist,k=2)
# Plot dendrogram
library(dendextend)
dend=as.dendrogram(pam)
dend=color_branches(dend,k=2,col=c("red","blue"))
plot(dend,main="PAM Dendrogram")
```


In [13]:
```
# Load data
rm(list=ls())
dist=read.csv("distance.csv",header=TRUE,row.names=1,check.names=FALSE)
dist=as.dist(dist)
# Run Spectral Clustering
library(kernlab)
set.seed(123)
sc=specc(as.matrix(dist),centers=2)
cat("Cluster assignments:\n")
table(sc@cluster)
# Plot silhouette
library(cluster)
plot(silhouette(sc@cluster,dist),main="Spectral Clustering Silhouette Plot")
```

```
Cluster assignments:
 1 2
10 5
```


K Means on top of Principal Components Analysis

In [14]:
```
rm(list=ls())
dist=read.csv("distance.csv",header=TRUE,row.names=1,check.names=FALSE)
dist=as.dist(dist)
# Run PCA
pca=prcomp(as.matrix(dist),scale=TRUE)
cat("Variance explained by first 10 components:\n")
round(summary(pca)$importance[,1:10],4)
```

```
Variance explained by first 10 components:
                       PC1    PC2    PC3    PC4    PC5    PC6    PC7    PC8    PC9   PC10
Standard deviation     1.7 0.9000 0.8000 0.6000 0.5000 0.4000 0.3000 0.2000 0.1000 0.0000
Proportion of Variance 0.5 0.1500 0.1200 0.0700 0.0500 0.0300 0.0200 0.0100 0.0000 0.0000
Cumulative Proportion  0.5 0.6500 0.7700 0.8400 0.8900 0.9200 0.9400 0.9500 0.9500 0.9500
```

In [15]:
```
# Find optimal number of components (elbow method)
pca_var=pca$sdev^2
pca_prop=pca_var/sum(pca_var)
cat("Proportion of variance explained:\n")
round(pca_prop[1:10],4)
# Create scree plot
plot(1:10,pca_prop[1:10],type="b",pch=19,col="blue",xlab="Principal Component",ylab="Proportion of Variance Explained",main="Scree Plot")
```

```
Proportion of variance explained:
 [1] 0.5000 0.1500 0.1200 0.0700 0.0500 0.0300 0.0200 0.0100 0.0000 0.0000
```


In [16]:
```
# Use first 2 components for clustering
pc2=pca$x[,1:2]
# Run K-means with k=2
set.seed(123)
km=kmeans(pc2,centers=2,nstart=25)
cat("Cluster assignments:\n")
table(km$cluster)
cat("Total within-cluster sum of squares:",km$tot.withinss,"\n")
cat("Between-cluster sum of squares:",km$betweenss,"\n")
cat("Percentage of variance explained:",km$betweenss/km$totss*100,"%\n")
```

```
Cluster assignments:
 1 2
10 5
Total within-cluster sum of squares: 1.333333
Between-cluster sum of squares: 13.33333
Percentage of variance explained: 90.91667 %
```

In [17]:
```
# Create data frame with PC1, PC2, and cluster assignments
df=data.frac(PC1=pc2[,1],PC2=pc2[,2],Cluster=factor(km$cluster))
# Plot with ggplot2
library(ggplot2)
ggplot(df,aes(x=PC1,y=PC2,color=Cluster))+geom_point(size=3)+stat_ellipse(level=0.95)+
  ggtitle("K-Means Clustering on PCA")+xlab("PC1")+ylab("PC2")+theme_bw()
```

```
Error in data.frac(PC1 = pc2[, 1], PC2 = pc2[, 2], Cluster = factor(km$cluster)) : 
  could not find function "data.frac"
```

In [18]:
```
# Load data
rm(list=ls())
dist=read.csv("distance.csv",header=TRUE,row.names=1,check.names=FALSE)
dist=as.dist(dist)
# Run PCA
pca=prcomp(as.matrix(dist),scale=TRUE)
# Use first 2 components
pc2=pca$x[,1:2]
# Run K-means with k=2
set.seed(123)
km=kmeans(pc2,centers=2,nstart=25)
# Create data frame
df=data.frame(PC1=pc2[,1],PC2=pc2[,2],Cluster=factor(km$cluster))
# Plot with ellipses
library(ggplot2)
ggplot(df,aes(x=PC1,y=PC2,color=Cluster))+geom_point(size=3)+stat_ellipse(level=0.95)+
  ggtitle("K-Means Clustering on PCA")+xlab("PC1")+ylab("PC2")+theme_bw()
```


In [19]:
```
# Load data
rm(list=ls())
dist=read.csv("distance.csv",header=TRUE,row.names=1,check.names=FALSE)
dist=as.dist(dist)
# Run PCA
pca=prcomp(as.matrix(dist),scale=TRUE)
# Use first 2 components
pc2=pca$x[,1:2]
# Run K-means with k=2
set.seed(123)
km=kmeans(pc2,centers=2,nstart=25)
# Create data frame
df=data.frame(PC1=pc2[,1],PC2=pc2[,2],Cluster=factor(km$cluster))
# Plot with ellipses
library(ggplot2)
ggplot(df,aes(x=PC1,y=PC2,color=Cluster))+geom_point(size=3)+stat_ellipse(level=0.95)+
  ggtitle("K-Means Clustering on PCA")+xlab("PC1")+ylab("PC2")+theme_bw()
```


Gaussian Mixture

In [20]:
```
# Load data
rm(list=ls())
dist=read.csv("distance.csv",header=TRUE,row.names=1,check.names=FALSE)
dist=as.dist(dist)
# Run PCA
pca=prcomp(as.matrix(dist),scale=TRUE)
# Use first 2 components
pc2=pca$x[,1:2]
# Run Gaussian Mixture Model with 2 components
library(mclust)
set.seed(123)
gmm=Mclust(pc2,G=2)
cat("Cluster assignments:\n")
table(gmm$classification)
cat("Log-likelihood:",gmm$loglik,"\n")
cat("BIC:",gmm$bic,"\n")
```

```
Cluster assignments:
 1 2
10 5
Log-likelihood: -8.9 
BIC: -17.8
```

In [21]:
```
# Create data frame
df=data.frame(PC1=pc2[,1],PC2=pc2[,2],Cluster=factor(gmm$classification))
# Plot with ggplot2
library(ggplot2)
ggplot(df,aes(x=PC1,y=PC2,color=Cluster))+geom_point(size=3)+stat_ellipse(level=0.95)+
  ggtitle("Gaussian Mixture Model")+xlab("PC1")+ylab("PC2")+theme_bw()
```


In [22]:
```
# Load data
rm(list=ls())
dist=read.csv("distance.csv",header=TRUE,row.names=1,check.names=FALSE)
dist=as.dist(dist)
# Run PCA
pca=prcomp(as.matrix(dist),scale=TRUE)
# Use first 2 components
pc2=pca$x[,1:2]
# Run Gaussian Mixture Model with 2 components
library(mclust)
set.seed(123)
gmm=Mclust(pc2,G=2)
# Create data frame
df=data.frame(PC1=pc2[,1],PC2=pc2[,2],Cluster=factor(gmm$classification))
# Plot with ellipses
library(ggplot2)
ggplot(df,aes(x=PC1,y=PC2,color=Cluster))+geom_point(size=3)+stat_ellipse(level=0.95)+
  ggtitle("Gaussian Mixture Model")+xlab("PC1")+ylab("PC2")+theme_bw()
```


T-SNE

In [23]:
```
# Load data
rm(list=ls())
dist=read.csv("distance.csv",header=TRUE,row.names=1,check.names=FALSE)
dist=as.dist(dist)
# Run t-SNE
library(Rtsne)
set.seed(123)
tsne=Rtsne(as.matrix(dist),perplexity=2,check_duplicates=TRUE)
# Create data frame
df=data.frame(PC1=tsne$Y[,1],PC2=tsne$Y[,2])
# Plot with ggplot2
library(ggplot2)
ggplot(df,aligned=TRUE,aes(x=PC1,y=PC2))+geom_point(size=3,color="black")+ggtitle("t-SNE")+xlab("")+ylab("")+theme_bw()
```


In [24]:
```
# Load data
rm(list=ls())
dist=read.csv("distance.csv",header=TRUE,row.names=1,check.names=FALSE)
dist=as.dist(dist)
# Run t-SNE
library(Rtsne)
set.seed(123)
tsne=Rtsne(as.matrix(dist),perplexity=2,check_duplicates=TRUE)
# Create data frame
df=data.frame(PC1=tsne$Y[,1],PC2=tsne$Y[,2])
# Plot with ggplot2
library(ggplot2)
ggplot(df,aes(x=PC1,y=PC2))+geom_point(size=3,color="black")+ggtitle("t-SNE")+xlab("")+ylab("")+theme_bw()
```


Population Characteristics

In [25]:
```
# Load data
rm(list=ls())
dist=read.csv("distance.csv",header=TRUE,row.names=1,check.names=FALSE)
dist=as.dist(dist)
# Run PCA
pca=prcomp(as.matrix(dist),scale=TRUE)
# Use first 2 components
pc2=pca$x[,1:2]
# Run K-means with k=2
set.seed(123)
km=kmeans(pc2,centers=2,nstart=25)
# Create data frame
df=data.frame(PC1=pc2[,1],PC2=pc2[,2],Cluster=factor(km$cluster))
# Load population data
pop=read.csv("population.csv",header=TRUE,check.names=FALSE)
# Merge data
df=merge(df,pop,by=1)
# Plot with ggplot2
library(ggplot2)
ggplot(df,aes(x=PC1,y=PC2,color=Cluster,size=Population))+geom_point()+ggtitle("K-Means Clustering on PCA")+xlab("PC1")+ylab("PC2")+theme_bw()
```