prediccion_con_kmeans-Copy1

January 6, 2021

0.1 Predicción de tipo de comunidad a partir del consumo energético utilizando *k-Means*

Alberto Ramos Sánchez

08/01/2021

0.1.1 Contenido

- Preparación de los datos
- A1: Predicción utilizando consumo eléctrico
  - A1.1: Predicción sin balancear los datos
  - A1.2: Predicción balanceando los datos
- A2: Predicción utilizando consumo de gas
  - A2.1: Predicción sin balancear los datos
  - A2.2: Predicción balanceando los datos

[1]:
```
import pandas as pd
import numpy as np

from sklearn.cluster import KMeans
from sklearn.metrics import accuracy_score
from sklearn import preprocessing

from dataloader import DataLoader

seed_val = 42
np.random.seed(seed_val)
import random
random.seed(seed_val)
```

[2]:
```
df = pd.read_csv('energy-usage-2010-clean.csv')
df
```

[2]:
```
       COMMUNITY AREA NAME  CENSUS BLOCK BUILDING TYPE BUILDING_SUBTYPE  \
0           Archer Heights  1.703157e+14   Residential        Multi < 7
1                  Ashburn  1.703170e+14   Residential         Multi 7+
2           Auburn Gresham  1.703171e+14    Commercial        Multi < 7
3                   Austin  1.703125e+14    Commercial        Multi < 7
```

```
4                Austin  1.703125e+14    Commercial       Multi < 7
…                     …             …             …               …
67046          Woodlawn  1.703184e+14   Residential   Single Family
67047          Woodlawn  1.703184e+14    Commercial       Multi < 7
67048          Woodlawn  1.703184e+14   Residential       Multi < 7
67049          Woodlawn  1.703184e+14   Residential   Single Family
67050          Woodlawn  1.703184e+14   Residential       Multi < 7

       KWH JANUARY 2010  KWH FEBRUARY 2010  KWH MARCH 2010  KWH APRIL 2010  \
0                   NaN                NaN             NaN             NaN
1                7334.0             7741.0          4214.0          4284.0
2                   NaN                NaN             NaN             NaN
3                   NaN                NaN             NaN             NaN
4                   NaN                NaN             NaN             NaN
…                     …                  …               …               …
67046            2705.0             1318.0          1582.0          1465.0
67047            1005.0             1760.0          1521.0          1832.0
67048            3567.0             3031.0          2582.0          2295.0
67049            1208.0             1055.0          1008.0          1109.0
67050            2717.0             3057.0          2695.0          3793.0

       KWH MAY 2010  KWH JUNE 2010  …  TOTAL POPULATION  TOTAL UNITS  \
0               NaN            NaN  …              89.0         24.0
1            2518.0         4273.0  …             112.0         67.0
2               NaN            NaN  …             102.0         48.0
3               NaN            NaN  …             121.0         56.0
4               NaN            NaN  …              62.0         23.0
…                 …              …  …                 …            …
67046        1494.0         2990.0  …             116.0         55.0
67047        2272.0         2361.0  …              31.0         24.0
67048        7902.0         4987.0  …              31.0         24.0
67049        1591.0         1367.0  …               0.0          0.0
67050        4237.0         5383.0  …              77.0         49.0

       AVERAGE STORIES  AVERAGE BUILDING AGE  AVERAGE HOUSESIZE  \
0                 2.00                 71.33               3.87
1                 2.00                 41.00               1.81
2                 3.00                 86.00               3.00
3                 2.00                 84.00               2.95
4                 2.00                 85.00               3.26
…                    …                     …                  …
67046             1.00                  0.00               3.14
67047             3.00                104.50               2.07
67048             2.33                100.67               2.07
67049             1.00                  0.00               0.00
67050             2.00                 79.40               2.57
```

```
       OCCUPIED UNITS  OCCUPIED UNITS PERCENTAGE  \
0                23.0                     0.9582
1                62.0                     0.9254
2                34.0                     0.7082
3                41.0                     0.7321
4                19.0                     0.8261
…                 …                        …
67046            37.0                     0.6727
67047            15.0                     0.6250
67048            15.0                     0.6250
67049             0.0                        NaN
67050            30.0                     0.6122

       RENTER-OCCUPIED HOUSING UNITS  RENTER-OCCUPIED HOUSING PERCENTAGE  \
0                                9.0                              0.3910
1                               50.0                              0.8059
2                               23.0                              0.6759
3                               32.0                              0.7800
4                               11.0                              0.5790
…                                …                                 …
67046                           26.0                              0.7030
67047                           13.0                              0.8670
67048                           13.0                              0.8670
67049                            0.0                                 NaN
67050                           28.0                              0.9329

       OCCUPIED HOUSING UNITS
0                        23.0
1                        62.0
2                        34.0
3                        41.0
4                        19.0
…                         …
67046                    37.0
67047                    15.0
67048                    15.0
67049                     0.0
67050                    30.0

[67051 rows x 73 columns]
```

[3]:
```
dl = DataLoader(df)
```

[4]:
```
dataset_energy = dl[dl.energy_cols + ['BUILDING TYPE']]
dataset_gas = dl[dl.gas_cols + ['BUILDING TYPE']]
```

Tenemos 3 clases: residencial, comercial e industrial

[5]:
```
dataset_energy['BUILDING TYPE'].unique().tolist()
```

[5]:
```
['Residential', 'Commercial', 'Industrial']
```

[6]:
```
dataset_gas['BUILDING TYPE'].unique().tolist()
```

[6]:
```
['Residential', 'Commercial', 'Industrial']
```

[7]:
```
dataset_energy.groupby(['BUILDING TYPE']).count()['KWH JANUARY 2010']
```

[7]:
```
BUILDING TYPE
Commercial     16630
Industrial        26
Residential    49447
Name: KWH JANUARY 2010, dtype: int64
```

[8]:
```
dataset_gas.groupby(['BUILDING TYPE']).count()['THERM JANUARY 2010']
```

[8]:
```
BUILDING TYPE
Commercial     14505
Industrial        31
Residential    46200
Name: THERM JANUARY 2010, dtype: int64
```

0.1.2 Preparación de los datos

[9]:
```
dataset_energy_X = dataset_energy[dl.energy_cols]
dataset_energy_y = dataset_energy['BUILDING TYPE']
```

[10]:
```
dataset_gas_X = dataset_gas[dl.gas_cols]
dataset_gas_y = dataset_gas['BUILDING TYPE']
```

Dividimos los datos en entrenamiento y test.

[11]:
```
dataset_energy_X_train = dataset_energy_X.sample(int(len(dataset_energy_X)*0.9))
dataset_energy_X_test = dataset_energy_X.drop(dataset_energy_X_train.index)

dataset_energy_y_train = dataset_energy_y[dataset_energy_X_train.index]
dataset_energy_y_test = dataset_energy_y.drop(dataset_energy_y_train.index)
```

[12]:
```
dataset_gas_X_train = dataset_gas_X.sample(int(len(dataset_gas_X)*0.9))
dataset_gas_X_test = dataset_gas_X.drop(dataset_gas_X_train.index)

dataset_gas_y_train = dataset_gas_y[dataset_gas_X_train.index]
dataset_gas_y_test = dataset_gas_y.drop(dataset_gas_y_train.index)
```

0.1.3 A1: Predicción utilizando consumo eléctrico

A1.1: Predicción sin balancear los datos

[13]:
```
scaler = preprocessing.StandardScaler()
data = dataset_energy_X_train
norm_data = scaler.fit(data).transform(data)

kmeans = KMeans(n_clusters = 3, random_state = 42)
kmeans.fit(norm_data)
```

[13]:
```
KMeans(n_clusters=3, random_state=42)
```

[14]:
```
cluster = kmeans.predict(norm_data)
```

Train score:

[15]:
```
d = {"Residential": 0, "Commercial": 1, "Industrial": 2}
accuracy_score(dataset_energy_y_train.map(d), cluster)
```

[15]:
```
0.7468062932831305
```

Test score:

[16]:
```
scaler = preprocessing.StandardScaler()
data = dataset_energy_X_test
norm_data = scaler.fit(data).transform(data)

cluster = kmeans.predict(norm_data)
```

[17]:
```
d = {"Residential": 0, "Commercial": 1, "Industrial": 2}
accuracy_score(dataset_energy_y_test.map(d), cluster)
```

[17]:
```
0.7597942822568446
```

A1.2: Predicción balanceando los datos Se balancean los datos.

[18]:
```
g = dataset_energy.groupby("BUILDING TYPE")
dataset_energy_bal = g.apply(lambda x: x.sample(g.size().min()).
 ↪reset_index(drop=True))

dataset_energy_bal["BUILDING TYPE"] = dataset_energy_bal.index
dataset_energy_bal["BUILDING TYPE"] = dataset_energy_bal["BUILDING TYPE"].
 ↪apply(lambda x: x[0])

dataset_energy_bal
```

[18]:
```
                    KWH JANUARY 2010  KWH FEBRUARY 2010  KWH MARCH 2010  \
BUILDING TYPE
Commercial    0               8076.0             7988.0          8598.0
              1              11412.0            10808.0         11285.0
              2                  0.0              719.0           274.0
```

```
             3          14581.0           13973.0         14613.0
             4              0.0               0.0             0.0
…                          …                 …             …
Residential  21          3345.0            2839.0          2537.0
             22          2054.0            1197.0          1005.0
             23          4187.0            3380.0          4229.0
             24          9582.0            7683.0          6774.0
             25          3587.0            4954.0          4450.0

                 KWH APRIL 2010  KWH MAY 2010  KWH JUNE 2010  KWH JULY 2010  \
BUILDING TYPE
Commercial   0           9562.0        9821.0        10288.0        11636.0
             1          12010.0       12872.0        14416.0        16091.0
             2            167.0         138.0          170.0          189.0
             3          11522.0       17511.0        19829.0        22480.0
             4              0.0         446.0          586.0          733.0
…                          …            …             …              …
Residential  21          3436.0        4591.0         5601.0         5700.0
             22          1744.0        2939.0         3136.0         3200.0
             23          3413.0        4593.0         9306.0         9454.0
             24          7880.0       11154.0        16201.0        17010.0
             25          4221.0        5379.0         6613.0         6679.0

                 KWH AUGUST 2010  KWH SEPTEMBER 2010  KWH OCTOBER 2010  \
BUILDING TYPE
Commercial   0           11290.0             10466.0           10062.0
             1           13222.0             11874.0           11062.0
             2             228.0               277.0             383.0
             3           19950.0             16179.0           11221.0
             4            1011.0               503.0             338.0
…                           …                   …                …
Residential  21           4033.0              2277.0            1965.0
             22           2205.0              1117.0             949.0
             23           5306.0              3826.0            3194.0
             24          12129.0              8166.0           11656.0
             25           6238.0              5310.0            7720.0

                 KWH NOVEMBER 2010  KWH DECEMBER 2010 BUILDING TYPE
BUILDING TYPE
Commercial   0             10598.0            11120.0    Commercial
             1             14382.0            14304.0    Commercial
             2              2139.0             1837.0    Commercial
             3             14346.0            15774.0    Commercial
             4              1916.0             1760.0    Commercial
…                             …                  …            …
Residential  21             4868.0             4094.0   Residential
             22             1521.0             2280.0   Residential
```

```
23            5763.0            7348.0   Residential
24           14434.0           14911.0   Residential
25            8025.0            8557.0   Residential

[78 rows x 13 columns]
```

[19]:
```
dataset_energy_bal_X = dataset_energy_bal[dl.energy_cols]
dataset_energy_bal_y = dataset_energy_bal['BUILDING TYPE']
```

[20]:
```
dataset_energy_bal_X_train = dataset_energy_bal_X.
 ↪sample(int(len(dataset_energy_bal_X)*0.9))
dataset_energy_bal_X_test = dataset_energy_bal_X.
 ↪drop(dataset_energy_bal_X_train.index)

dataset_energy_bal_y_train = dataset_energy_bal_y[dataset_energy_bal_X_train.
 ↪index]
dataset_energy_bal_y_test = dataset_energy_bal_y.
 ↪drop(dataset_energy_bal_y_train.index)
```

Predicción

[21]:
```
scaler = preprocessing.StandardScaler()
data = dataset_energy_bal_X_train
norm_data = scaler.fit(data).transform(data)

kmeans = KMeans(n_clusters = 3, random_state = 42)
kmeans.fit(norm_data)
```

[21]:
```
KMeans(n_clusters=3, random_state=42)
```

[22]:
```
cluster = kmeans.predict(norm_data)
```

Train score:

[23]:
```
d = {"Residential": 0, "Commercial": 1, "Industrial": 2}
accuracy_score(dataset_energy_bal_y_train.map(d), cluster)
```

[23]:
```
0.37142857142857144
```

Test score:

[24]:
```
scaler = preprocessing.StandardScaler()
data = dataset_energy_bal_X_test
norm_data = scaler.fit(data).transform(data)

cluster = kmeans.predict(norm_data)
```

[25]:
```
d = {"Residential": 0, "Commercial": 1, "Industrial": 2}
accuracy_score(dataset_energy_bal_y_test.map(d), cluster)
```

[25]: 0.5

0.1.4 A2: Predicción utilizando consumo de gas

A2.1: Predicción sin balancear los datos

[26]:
```
scaler = preprocessing.StandardScaler()
data = dataset_gas_X_train
norm_data = scaler.fit(data).transform(data)

kmeans = KMeans(n_clusters = 3, random_state = 42)
kmeans.fit(norm_data)
```

[26]: KMeans(n_clusters=3, random_state=42)

[27]:
```
cluster = kmeans.predict(norm_data)
```

Train score:

[28]:
```
d = {"Residential": 0, "Commercial": 1, "Industrial": 2}
accuracy_score(dataset_gas_y_train.map(d), cluster)
```

[28]: 0.7603819838278878

Test score:

[29]:
```
scaler = preprocessing.StandardScaler()
data = dataset_gas_X_test
norm_data = scaler.fit(data).transform(data)

cluster = kmeans.predict(norm_data)
```

[30]:
```
d = {"Residential": 0, "Commercial": 1, "Industrial": 2}
accuracy_score(dataset_gas_y_test.map(d), cluster)
```

[30]: 0.7627593019427066

A2.2: Predicción balanceando los datos Se balancean los datos.

[31]:
```
g = dataset_gas.groupby("BUILDING TYPE")
dataset_gas_bal = g.apply(lambda x: x.sample(g.size().min()).
 ↪reset_index(drop=True))

dataset_gas_bal["BUILDING TYPE"] = dataset_gas_bal.index
dataset_gas_bal["BUILDING TYPE"] = dataset_gas_bal["BUILDING TYPE"].
 ↪apply(lambda x: x[0])

dataset_gas_bal
```

[31]:

```
                          THERM JANUARY 2010  THERM FEBRUARY 2010  THERM MARCH 2010  \
BUILDING TYPE
Commercial     0                      2038.0               1694.0            1699.0
               1                      4909.0               4045.0            2223.0
               2                       502.0                462.0             487.0
               3                      1686.0               1397.0            1519.0
               4                      3405.0               4301.0            3137.0
…                                        …                    …                 …
Residential    26                     3500.0               3353.0            2445.0
               27                     3830.0               3445.0            2646.0
               28                     6662.0               5718.0            4056.0
               29                     5927.0               5112.0            4867.0
               30                     4426.0               3886.0            2871.0

                          THERM APRIL 2010  THERM MAY 2010  THERM JUNE 2010  \
BUILDING TYPE
Commercial     0                    1319.0           469.0            368.0
               1                     698.0           291.0            261.0
               2                     721.0           755.0            618.0
               3                    2158.0           590.0           1240.0
               4                     888.0           425.0             94.0
…                                      …               …                …
Residential    26                   1249.0           773.0            368.0
               27                   1422.0           944.0            539.0
               28                   2037.0          1070.0            569.0
               29                   2359.0          1544.0           1126.0
               30                   1643.0           962.0            380.0

                          THERM JULY 2010  THERM AUGUST 2010  THERM SEPTEMBER 2010  \
BUILDING TYPE
Commercial     0                    256.0              207.0                 209.0
               1                    260.0               90.0                  79.0
               2                    862.0              610.0                 746.0
               3                    602.0             1233.0                 536.0
               4                     39.0               28.0                  26.0
…                                     …                  …                     …
Residential    26                   273.0              269.0                 257.0
               27                   356.0              368.0                 355.0
               28                   482.0              449.0                 606.0
               29                   857.0              890.0                 807.0
               30                   379.0              314.0                 323.0

                          THERM OCTOBER 2010  THERM NOVEMBER 2010  \
BUILDING TYPE
Commercial     0                       217.0                605.0
               1                       162.0               1365.0
               2                       604.0                547.0
```

```
                3              1407.0               1291.0
                4                74.0                114.0
…                                …                  …
Residential     26              335.0               1014.0
                27              447.0               1111.0
                28             1099.0               2390.0
                29              972.0               1352.0
                30              785.0               1778.0

                    THERM DECEMBER 2010 BUILDING TYPE
BUILDING TYPE
Commercial      0               1199.0    Commercial
                1               4711.0    Commercial
                2                636.0    Commercial
                3               2108.0    Commercial
                4               2017.0    Commercial
…                                …            …
Residential     26              2637.0   Residential
                27              2884.0   Residential
                28              7094.0   Residential
                29              2864.0   Residential
                30              4090.0   Residential

[93 rows x 13 columns]
```

[32]:
```
dataset_gas_bal_X = dataset_gas_bal[dl.gas_cols]
dataset_gas_bal_y = dataset_gas_bal['BUILDING TYPE']
```

[33]:
```
dataset_gas_bal_X_train = dataset_gas_bal_X.sample(int(len(dataset_gas_bal_X)*0.
 ↪9))
dataset_gas_bal_X_test = dataset_gas_bal_X.drop(dataset_gas_bal_X_train.index)

dataset_gas_bal_y_train = dataset_gas_bal_y[dataset_gas_bal_X_train.index]
dataset_gas_bal_y_test = dataset_gas_bal_y.drop(dataset_gas_bal_y_train.index)
```

Predicción

[34]:
```
scaler = preprocessing.StandardScaler()
data = dataset_gas_bal_X_train
norm_data = scaler.fit(data).transform(data)

kmeans = KMeans(n_clusters = 3, random_state = 42)
kmeans.fit(norm_data)
```

[34]:
```
KMeans(n_clusters=3, random_state=42)
```

[35]:
```
cluster = kmeans.predict(norm_data)
```

Train score:

[36]:
```
d = {"Residential": 0, "Commercial": 1, "Industrial": 2}
accuracy_score(dataset_gas_bal_y_train.map(d), cluster)
```

[36]: 0.3614457831325301

Test score:

[37]:
```
scaler = preprocessing.StandardScaler()
data = dataset_gas_bal_X_test
norm_data = scaler.fit(data).transform(data)

cluster = kmeans.predict(norm_data)
```

[38]:
```
d = {"Residential": 0, "Commercial": 1, "Industrial": 2}
accuracy_score(dataset_gas_bal_y_test.map(d), cluster)
```

[38]: 0.1