6_Life_Expectancy_Recursive_Feature_Elimination

January 29, 2023

Table of Contents

1 Life_Expectancy_WHO_UN_Analysis_Modeling

1.1 Recursive_Feature_Elimination

To: Magnimind

From: Matt Curcio, matt.curcio.ri@gmail.com

Date: 2023-01-29

Re: NOTEBOOK #6

- **Use** `Clean_LE_Data_FEng_4.csv`

[1]:

```python
# Common Python Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

# import warnings
import warnings
warnings.filterwarnings("ignore")

# Libraries from Sklearn
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression

import statsmodels.api as sm

from statsmodels.stats.outliers_influence import variance_inflation_factor
```

[2]:

```
!ls *.csv
```

```
Clean_LE_Data_FEng_4.csv      Life_Expectancy_Data.csv  y_test.csv
Clean_LE_Data_Post_EDA_3.csv  x_test.csv                y_train.csv
Clean_LE_Data_w_Means_2.csv   x_train.csv
```

[3]:

```
# Reality check ;)

df = pd.read_csv("Clean_LE_Data_FEng_4.csv", header=0)

# Convert 4 vars to categorical
df['Country'] = pd.Categorical(df['Country'])
df['Year'] = pd.Categorical(df['Year'])
df['Status'] = pd.Categorical(df['Status'])
df['Region'] = pd.Categorical(df['Region'])

df.head(3)
```

[3]:

```
       Country  Year Status  LifeExpectancy  AdultMort  EtOH  PercExpen  \
0  Afghanistan  2015      0            65.0      263.0  0.01  71.279624
1  Afghanistan  2014      0            59.9      271.0  0.01  73.523582
2  Afghanistan  2013      0            59.9      268.0  0.01  73.219243

   Measles   BMI  lt5yD  Polio  TotalExpen   DTP  HIV  Thin1_19y  Income  \
0     1154  19.1     83    6.0        8.16  65.0  0.1       17.2   0.479
1      492  18.6     86   58.0        8.18  62.0  0.1       17.5   0.476
2      430  18.1     89   62.0        8.13  64.0  0.1       17.7   0.470

   Education Region
0       10.1      2
1       10.0      2
2        9.9      2
```

1.2 Recursive Feature Elimination

1.2.1 train_test_split Section

[4]:

```
x = df.drop(['LifeExpectancy','Country'], axis=1)
y = df['LifeExpectancy']
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y,
                                                    test_size=0.3,
                                                    random_state=100)

print('\nShape of x_train is', {x_train.shape})
print('\nShape of y_train is', {y_train.shape})

print('\nShape of x_test is', {x_test.shape})
print('\nShape of y_test is', {y_test.shape})
```

```
Shape of x_train is {(2049, 16)}

Shape of y_train is {(2049,)}

Shape of x_test is {(879, 16)}

Shape of y_test is {(879,)}
```

1.2.2 NOTE 1:

- Cannot Use `Stratify`, option for `train_test_split`. There are not sufficient class labels of one of your classes to keep the data splitting ratio equal to test_size.

[5]:

```
x_train
```

[5]:

```
      Year Status  AdultMort   EtOH    PercExpen  Measles   BMI  lt5yD  Polio  \
1346  2013      0       28.0   6.48    26.407266       73  51.4      6   98.0
2073  2006      0       83.0   1.28   448.595299      144  65.0      0   95.0
746   2005      1       92.0  11.28  7627.412444        2  55.0      0   93.0
2667  2004      0       15.0   1.36   379.765905        1  51.4      4   97.0
348   2003      0      693.0   5.51   299.367125       59  31.6      4   96.0
…     …    …          …     …          …         …   …      …    …
1930  2005      0      118.0   0.63   854.011576       25  49.5      1   99.0
79    2000      0      156.0   7.27  1127.743470        0  38.2      0   96.0
1859  2012      0      157.0   3.63   334.817425        0  51.7      3   99.0
2840  2007      0      157.0   1.01   342.490856        0  46.6      0   67.0
1544  2007      1       82.0  11.75   267.051312        0  57.9      0   96.0

      TotalExpen   DTP   HIV  Thin1_19y  Income  Education Region
1346        4.30  98.0   0.1        2.4   0.782       15.0      1
2073        2.58  96.0   0.1        4.9   0.835       13.7      2
746         9.77  93.0   0.1        1.2   0.897       16.6      5
2667        5.63  97.0   0.1        6.4   0.673       13.6      2
348         4.65  96.0  31.9        1.9   0.567       11.8      7
…          …   …    …        …       …      …   …
1930        2.59  99.0   0.1        7.0   0.742       11.5      2
79          4.13  95.0   0.1        3.7   0.000        0.0      8
```

```
1859        8.40  98.0   0.1        1.8   0.625       11.5       8
2840        3.67  67.0   0.1        1.6   0.579       10.7       6
1544        6.80  99.0   0.1        0.9   0.877       13.5       5

[2049 rows x 16 columns]
```

[6]:

```python
y_train
```

[6]:

```
1346    69.5
2073    76.6
746     78.1
2667    74.0
348     46.4
        …
1930    74.3
79      73.6
1859    73.9
2840    73.0
1544    79.7
Name: LifeExpectancy, Length: 2049, dtype: float64
```

[7]:

```python
x_train.to_csv('x_train.csv', index=False)

x_test.to_csv('x_test.csv', index=False)

y_train.to_csv('y_train.csv', index=False)

y_test.to_csv('y_test.csv', index=False)
```

[8]:

```python
scaler = MinMaxScaler()

numerical_vars = ['AdultMort','EtOH','PercExpen',
                  'Measles','BMI','lt5yD','Polio','TotalExpen',
                  'DTP','HIV','Thin1_19y','Income','Education']

x_train[numerical_vars] = scaler.fit_transform(x_train[numerical_vars])
```

[9]:

```python
x_train
```

[9]:

```
      Year Status  AdultMort      EtOH  PercExpen   Measles       BMI   lt5yD  \
1346  2013       0   0.037396  0.362262   0.001356  0.000400  0.657963  0.0024
2073  2006       0   0.113573  0.071109   0.023029  0.000789  0.835509  0.0000
746   2005       1   0.126039  0.631019   0.391553  0.000011  0.704961  0.0000
2667  2004       0   0.019391  0.075588   0.019495  0.000005  0.657963  0.0016
348   2003       0   0.958449  0.307951   0.015368  0.000323  0.399478  0.0016
…    …    …        …       …        …       …       …     …
1930  2005       0   0.162050  0.034714   0.043841  0.000137  0.633159  0.0004
```

```
79    2000       0   0.214681  0.406495   0.057893  0.000000  0.485640  0.0000
1859  2012       0   0.216066  0.202688   0.017188  0.000000  0.661880  0.0012
2840  2007       0   0.216066  0.055991   0.017582  0.000000  0.595300  0.0000
1544  2007       1   0.112188  0.657335   0.013709  0.000000  0.742820  0.0000

         Polio  TotalExpen       DTP       HIV  Thin1_19y    Income  \
1346  0.989583    0.211151  0.989691  0.000000   0.083333  0.827513
2073  0.958333    0.109134  0.969072  0.000000   0.173913  0.883598
746   0.937500    0.535587  0.938144  0.000000   0.039855  0.949206
2667  0.979167    0.290036  0.979381  0.000000   0.228261  0.712169
348   0.968750    0.231910  0.969072  0.629703   0.065217  0.600000
…        …         …       …       …        …       …
1930  1.000000    0.109727  1.000000  0.000000   0.250000  0.785185
79    0.968750    0.201068  0.958763  0.000000   0.130435  0.000000
1859  1.000000    0.454330  0.989691  0.000000   0.061594  0.661376
2840  0.666667    0.173784  0.670103  0.000000   0.054348  0.612698
1544  0.968750    0.359431  1.000000  0.000000   0.028986  0.928042

      Education Region
1346   0.724638      1
2073   0.661836      2
746    0.801932      5
2667   0.657005      2
348    0.570048      7
…         …    …
1930   0.555556      2
79     0.000000      8
1859   0.555556      8
2840   0.516908      6
1544   0.652174      5

[2049 rows x 16 columns]
```

1.2.3 RFE fitting

[10]:
```
lm = LinearRegression()
lm.fit(x_train,y_train)
```

[10]:
```
LinearRegression()
```

[11]:
```
rfe = RFE(lm)

rfe = rfe.fit(x_train, y_train)
```

[12]:
```
feature_importance = list(zip(x_train.columns,rfe.support_,rfe.ranking_))
```

[13]:

```python
def Sort_Tuple(tup):
    """ reverse = None (Sorts in Ascending order)
    key is set to sort using second element of
    sublist lambda has been used
    """
    tup.sort(key = lambda x: x[2])
    return tup


# printing the sorted list of tuples
Sort_Tuple(feature_importance)
```

[13]:

```
[('AdultMort', True, 1),
 ('PercExpen', True, 1),
 ('Measles', True, 1),
 ('BMI', True, 1),
 ('DTP', True, 1),
 ('HIV', True, 1),
 ('Income', True, 1),
 ('Education', True, 1),
 ('Polio', False, 2),
 ('Thin1_19y', False, 3),
 ('Status', False, 4),
 ('lt5yD', False, 5),
 ('TotalExpen', False, 6),
 ('EtOH', False, 7),
 ('Region', False, 8),
 ('Year', False, 9)]
```

1.2.4 NOTE 2: Inference

- **USE** For first model: `Income`, `Education`,`HIV`, `DTP`, `Polio`, `lt5y_D`, `AdultMort`

[ ]:

```python

```