L2_ROA_EEA

February 12, 2023

1 Carga de base de datos

- Uso de libreria pandas
- Base de datos de la EEA (Encuesta Economica Anual)
- Construir el indicador por empresa de ROA

El indicador es calculado como:

ROA = Beneficio Neto obtenido/ Activo total de una empresa

- Definicion: utilidad que recibe la empresa por cada sol(dolar) invertido en sus distintos bienes y de los cuales se espera que generen ganancias a futuro
- ROA se entiende como el retorno que da la inversión que hace una empresa
- El valor del calculo estara en decimal , asi que se multiplicara por 100 para tenerlo en porcentaje

1.1 Carga de Librerias

[1]:
```
# Instalar la informacion de python
#!pip install pyreadstat
```

[2]:
```
import pandas as pd
import numpy as np
import os
import sys
import pyreadstat
```

1.2 Base de Activos (Estados financieros)

[3]:
```
ruta = 'D:/Dropbox/BASES/INEI-EEA/DATA/2018/Download/630-Modulo1570/
 ↪a2017_s11_fD2'
os.chdir(ruta)
os.getcwd()
```

[3]:
```
'D:\\Dropbox\\BASES\\INEI-
EEA\\DATA\\2018\\Download\\630-Modulo1570\\a2017_s11_fD2'
```

[4]:
```
data = pd.read_spss('a2017_s11_fD2_c02_1.sav')
data.shape
#/content/Data/a2019_s11_fD2_c00_1.sav
```

[4]:
```
(51362, 10)
```

[5]:
```
data.head(2)
```

[5]:
```
          IRUC Nroestablec CodSector CodFormato CodCapitulo  \
0  00000009996         000        11         D2          02
1  00000009996         000        11         D2          02

  FlagEstablecimiento Clave        P01         P02 FACTOR_EXP
0                   1   001  2653557.0  10082719.0  2.1666667
1                   1   002        0.0         0.0  2.1666667
```

[6]:
```
# Tipo de variables
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51362 entries, 0 to 51361
Data columns (total 10 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   IRUC                 51362 non-null  object
 1   Nroestablec          51362 non-null  object
 2   CodSector            51362 non-null  object
 3   CodFormato           51362 non-null  object
 4   CodCapitulo          51362 non-null  object
 5   FlagEstablecimiento  51362 non-null  object
 6   Clave                51362 non-null  object
 7   P01                  51362 non-null  float64
 8   P02                  51362 non-null  float64
 9   FACTOR_EXP           51362 non-null  object
dtypes: float64(2), object(8)
memory usage: 3.9+ MB
```

[7]:
```
data.groupby(['Clave'])['P01'].describe()
```

[7]:
```
       count          mean           std          min          25%  \
Clave
001    842.0  5.222453e+06  2.303870e+07          0.0    306031.00
002    842.0  2.461691e+05  2.982536e+06          0.0         0.00
003    842.0  1.930160e+07  4.153179e+07          0.0   2213156.00
```

```
004     842.0  5.190767e+05  2.135042e+06        -141.0       2113.25
005     842.0  5.129626e+06  2.197327e+07           0.0      70921.00
…       …            …            …            …          …
057     842.0  2.509488e+07  1.212425e+08 -124488269.0     807209.00
058     842.0  1.984316e+07  9.752589e+07 -134320530.0      35117.50
059     842.0  5.251716e+06  2.712387e+07  -33675827.0     179857.75
060     842.0  6.593409e+07  2.932213e+08  -38313031.0    5323666.25
061     842.0  1.280389e+08  4.858309e+08     588763.0  13141252.25

              50%          75%           max
Clave
001      998470.0   3113109.50  4.128284e+08
002           0.0         0.00  7.505180e+07
003     5329554.0  16078721.50  5.459283e+08
004       42564.0    281402.25  4.243648e+07
005      423897.0   2152754.25  3.947218e+08
…             …           …            …
057     3031028.0  13890516.75  2.300609e+09
058     1918173.5   9769397.50  1.834452e+09
059      857543.0   2965916.50  4.661577e+08
060    12173514.5  44605449.50  6.171574e+09
061    28785077.5  87513721.00  8.581819e+09

[61 rows x 8 columns]
```

[8]:
```python
#data[['CLAVE']]
#tab = data.groupby(['CLAVE', 'dato1']).size()
tab = data.groupby(['Clave']).size()
tab
```

[8]:
```
Clave
001     842
002     842
003     842
004     842
005     842
       …
057     842
058     842
059     842
060     842
061     842
Length: 61, dtype: int64
```

[9]:
```python
# Filtrando la base de datos
# Total activo (A+B) 30 en el 2017
# Resultado del Ejercicio 59 (engañoso)
```

```
data_nueva = data[(data.Clave=="030") | (data.Clave=="059")]
data_nueva.shape
```

```
[9]: (1684, 10)
```

```
[10]: data_nueva.head(4)
```

```
[10]:             IRUC Nroestablec CodSector CodFormato CodCapitulo  \
      11   00000009996         000        11         D2          02
      42   00000009996         000        11         D2          02
      83   00000013896         000        11         D2          02
      114  00000013896         000        11         D2          02

          FlagEstablecimiento Clave         P01         P02 FACTOR_EXP
      11                    1   059   1014691.0   3142945.0  2.1666667
      42                    1   030  37071421.0  37802013.0  2.1666667
      83                    1   059  -4309016.0  -4371133.0  2.1666667
      114                   1   030  19322966.0  39218937.0  2.1666667
```

```
[11]: # Pasar de un formato long a wide: comando pivot
      #data_ef = data_nueva.pivot(index=('IRUC','NroEstablec','CodSector'),␣
       ↪columns='CLAVE', values='dato1' )
      data_ef = data_nueva[(data_nueva.Clave=='030')]
      data_ef.shape
```

```
[11]: (842, 10)
```

```
[12]: data_ef.head(4)
```

```
[12]:             IRUC Nroestablec CodSector CodFormato CodCapitulo  \
      42   00000009996         000        11         D2          02
      114  00000013896         000        11         D2          02
      186  00000010325         000        11         D2          02
      258  00000011609         000        11         D2          02

          FlagEstablecimiento Clave         P01         P02 FACTOR_EXP
      42                    1   030  37071421.0  37802013.0  2.1666667
      114                   1   030  19322966.0  39218937.0  2.1666667
      186                   1   030  91801660.0  82226939.0  2.1666667
      258                   1   030  17440446.0  15445112.0  2.1666667
```

```
[13]: data_ef['Activos'] = data_ef['P01']
      data_ef = data_ef[['IRUC','CodSector','Activos']]
      data_ef['Activos'] = data_ef['Activos'] / 1000000
      data_ef.sort_values('Activos')
```

```
C:\Users\edinson\AppData\Local\Temp\ipykernel_5636\3582557537.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data_ef['Activos'] = data_ef['P01']
```

[13]:
```
              IRUC CodSector      Activos
40552  00000048544        11     0.588763
26474  00000110724        11     1.482189
37231  00000145797        11     1.699023
44038  00000145798        11     2.216882
44380  00000011045        11     2.315611
…              …       …           …
12876  00000017765        11  3102.797426
26844  00000016567        11  3198.962517
16831  00000011050        11  3329.293531
12815  00000015538        11  7811.421454
12084  00000010223        11  8581.819213

[842 rows x 3 columns]
```

[14]:
```
data_ef.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 842 entries, 42 to 51343
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   IRUC       842 non-null    object
 1   CodSector  842 non-null    object
 2   Activos    842 non-null    float64
dtypes: float64(1), object(2)
memory usage: 26.3+ KB
```

[15]:
```
data_ef.groupby(['CodSector'])['Activos'].describe()
```

[15]:
```
           count        mean         std       min        25%        50%  \
CodSector
11         842.0  128.038916  485.830901  0.588763  13.141252  28.785077

                 75%          max
CodSector
11         87.513721  8581.819213
```

1.3 Base de utilidad (Estado de Resultados)

[16]:
```
base = pd.read_spss('a2017_s11_fD2_c03_1.sav')
base.info()
base.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51362 entries, 0 to 51361
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   IRUC                51362 non-null  object
 1   Nroestablec         51362 non-null  object
 2   CodSector           51362 non-null  object
 3   CodFormato          51362 non-null  object
 4   CodCapitulo         51362 non-null  object
 5   FlagEstablecimiento 51362 non-null  object
 6   Clave               51362 non-null  object
 7   P01                 51362 non-null  float64
 8   FACTOR_EXP          51362 non-null  object
dtypes: float64(1), object(8)
memory usage: 3.5+ MB
```

[16]:
```
           IRUC Nroestablec CodSector CodFormato CodCapitulo  \
0  00000016131         000        11         D2          03
1  00000016131         000        11         D2          03
2  00000016131         000        11         D2          03
3  00000016131         000        11         D2          03
4  00000016131         000        11         D2          03

  FlagEstablecimiento Clave          P01 FACTOR_EXP
0                   1   001          0.0  1.1343284
1                   1   002          0.0  1.1343284
2                   1   003          0.0  1.1343284
3                   1   004          0.0  1.1343284
4                   1   005  197193646.0  1.1343284
```

[17]:
```
base_er = base[(base.Clave =='061' )]
base_er['Utility'] = base_er['P01']
base_er['Utility'] = base_er['Utility'] / 1000000
base_er = base_er[['IRUC','CodSector','Clave','Utility']]
base_er
```

```
C:\Users\edinson\AppData\Local\Temp\ipykernel_5636\2080168219.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  base_er['Utility'] = base_er['P01']
C:\Users\edinson\AppData\Local\Temp\ipykernel_5636\2080168219.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  base_er['Utility'] = base_er['Utility'] / 1000000
```

[17]:
```
                IRUC CodSector Clave   Utility
6       00000016131        11   061  7.581764
67      00000030417        11   061  1.091191
88      00000009960        11   061 -5.398127
148     00000010524        11   061  0.674295
168     00000124984        11   061  0.290803
…               …          …    …         …
51063   00000113173        11   061  0.460107
51124   00000029153        11   061  1.606037
51185   00000016406        11   061  0.502365
51246   00000120743        11   061  1.264818
51304   00000033560        11   061  0.405350

[842 rows x 4 columns]
```

[18]:
```python
print(base_er)
```

```
                IRUC CodSector Clave   Utility
6       00000016131        11   061  7.581764
67      00000030417        11   061  1.091191
88      00000009960        11   061 -5.398127
148     00000010524        11   061  0.674295
168     00000124984        11   061  0.290803
…               …          …    …         …
51063   00000113173        11   061  0.460107
51124   00000029153        11   061  1.606037
51185   00000016406        11   061  0.502365
51246   00000120743        11   061  1.264818
51304   00000033560        11   061  0.405350

[842 rows x 4 columns]
```

[19]:
```python
base_er.sort_values('Utility')
```

```
[19]:                IRUC CodSector Clave     Utility
      48719  00000122510        11   061  -33.675827
      31973  00000010756        11   061  -27.165397
      27214  00000011052        11   061  -26.340844
      34171  00000032901        11   061  -24.985856
      24132  00000027390        11   061  -24.643831
      …              …        …    …          …
      14499  00000018708        11   061  141.060905
      32894  00000011050        11   061  163.655989
      41818  00000016567        11   061  190.781272
      49325  00000015538        11   061  449.616213
      27072  00000010223        11   061  466.157664

      [842 rows x 4 columns]
```

1.4 Uniendo ambas bases de datos

```
[20]: data_ef
```

```
[20]:                IRUC CodSector     Activos
      42     00000009996        11   37.071421
      114    00000013896        11   19.322966
      186    00000010325        11   91.801660
      258    00000011609        11   17.440446
      330    00000009959        11  344.273677
      …              …        …          …
      51099  00000145794        11    5.804285
      51144  00000016375        11   10.077662
      51221  00000013759        11    7.741159
      51282  00000017061        11    7.723469
      51343  00000033329        11    6.759145

      [842 rows x 3 columns]
```

```
[21]: base_er
```

```
[21]:                IRUC CodSector Clave   Utility
      6      00000016131        11   061  7.581764
      67     00000030417        11   061  1.091191
      88     00000009960        11   061 -5.398127
      148    00000010524        11   061  0.674295
      168    00000124984        11   061  0.290803
      …              …        …    …        …
      51063  00000113173        11   061  0.460107
      51124  00000029153        11   061  1.606037
      51185  00000016406        11   061  0.502365
```

```
51246  00000120743        11   061  1.264818
51304  00000033560        11   061  0.405350

[842 rows x 4 columns]
```

[22]:
```
result =pd.merge(data_ef, base_er, how='inner')
result.shape
```

[22]:
```
(842, 5)
```

[23]:
```
result.head(3)
```

[23]:
```
          IRUC CodSector    Activos Clave   Utility
0  00000009996        11  37.071421   061  1.014691
1  00000013896        11  19.322966   061 -4.309016
2  00000010325        11  91.801660   061 -8.041010
```

[24]:
```
result['Utility'].describe()
```

[24]:
```
count    842.000000
mean       5.251716
std       27.123865
min      -33.675827
25%        0.179858
50%        0.857543
75%        2.965916
max      466.157664
Name: Utility, dtype: float64
```

1.5 Calculo del ROA

[25]:
```
result['ROA'] = (result['Utility'] / result['Activos'] )*100
result.head(4)
```

[25]:
```
          IRUC CodSector    Activos Clave   Utility        ROA
0  00000009996        11  37.071421   061  1.014691   2.737125
1  00000013896        11  19.322966   061 -4.309016 -22.299972
2  00000010325        11  91.801660   061 -8.041010  -8.759112
3  00000011609        11  17.440446   061  0.619705   3.553263
```

[26]:
```
result['ROA'].describe()
```

[26]:
```
count    842.000000
mean       4.412435
std        8.768000
min      -84.816916
```

```
25%         0.832017
50%         3.452078
75%         6.966020
max        55.820832
Name: ROA, dtype: float64
```

[27]:
```
result.groupby(['CodSector'])['Activos','Utility','ROA'].describe()
```

```
C:\Users\edinson\AppData\Local\Temp\ipykernel_5636\2409138246.py:1:
FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of
keys) will be deprecated, use a list instead.
  result.groupby(['CodSector'])['Activos','Utility','ROA'].describe()
```

[27]:
```
           Activos                                                        \
             count        mean         std       min        25%        50%
CodSector
11           842.0  128.038916  485.830901  0.588763  13.141252  28.785077

                                  Utility          …            \
                75%          max    count      mean …       75%
CodSector                                          …
11          87.513721  8581.819213   842.0  5.251716 …  2.965916

                         ROA                                              \
                 max  count     mean    std         min       25%       50%
CodSector
11          466.157664  842.0  4.412435  8.768 -84.816916  0.832017  3.452078


               75%        max
CodSector
11          6.96602  55.820832

[1 rows x 24 columns]
```

1.6 Exportando hacia excel

[29]:
```
# Export to excel
#result.to_excel('/content/Data/ROA_2019.xlsx')
output = 'D:/Dropbox/BASES/ENAHO/Python_scripts'
#OneDrive - Pacífico Compañía de Seguros y Reaseguros/Edinson_C/19.
 ↪Universidades/759-Modulo05/Enaho01A-2021-500.sav
#ruta
# Se cambia la informacion de la ruta con el comando: os.chdir()
os.chdir(output)
os.getcwd()
```

```python
result.to_csv('ROA_2019.csv')
result.to_excel('ROA_2019.xlsx')
```

[ ]:

```python

```