Questions

September 29, 2019

1 Data Science Challenge

[11]:
```
# If you'd like to install packages that aren't installed by default, uncomment
 ↪the last two lines of this cell and replace <package list> with a list of
 ↪your packages.
# This will ensure your notebook has all the dependencies and works everywhere

#import sys
#!{sys.executable} -m pip install <package list>
```

[21]:
```
#Libraries
import pandas as pd
import numpy as np
pd.set_option("display.max_columns", 101)
```

1.1 Data Description

Column	Description
surface_area	The total area in square kilometers
agricultural_land	The agricultural land of the total area in square kilometers
forest_area	The forest area in the total area in square kilometers
armed_forces_total	The count of armed forces paid by this geographical area
urban_pop_major_cities	The percent of the total population dwelling in major cities
urban_pop_minor_cities	The percent of the total population dwelling in minor cities
national_income	National Income as an ordinal categorical variable
inflation_annual	Yearly Inflation Rate
inflation_monthly	Average Monthly Inflation Rate = annual inflation/12
inflation_weekly	Average Weekly Inflation Rate = annual inflation/52

Column	Description
mobile_subscriptions	Describes the number of mobile subscriptions per person
internet_users	The average number of people using the internet in a range of 100 or 1000 people
secure_internet_servers_total	The actual number of secure internet servers in the area
improved_sanitation	The known access of the population to improved sanitation facilities
women_parliament_seats_rate	Describes the percent range of parliament seats occupied by women
life_expectancy	Years of life an average person is expected to live in this area (target variable).

1.2 Data Wrangling & Visualization

[13]:
```
# Dataset is already loaded below
data = pd.read_csv("train.csv",index_col=0)
```

[14]:
```
data.head()
```

[14]:
```
   surface_area  agricultural_land   forest_area  armed_forces_total  \
0      120540.0       2.632839e+06  5.417843e+06           1379000.0
1      752610.0       2.403039e+07  4.957554e+07             16500.0
2        1396.0       3.000000e+03  8.000000e+01                 NaN
3     2758812.0       1.228845e+08  1.050943e+08           1518650.0
4         340.0       1.100000e+04  1.699000e+04                 NaN

   urban_pop_major_cities  urban_pop_minor_cities national_income  \
0               55.747169                4.688831         unknown
1               16.890687               23.136313        very low
2               18.390090               23.139910         unknown
3               50.966885               24.522427            high
4                5.311885               30.271115         unknown

   inflation_annual  inflation_monthly  inflation_weekly  \
0               NaN                NaN               NaN
1               NaN           0.581473               NaN
2               NaN                NaN               NaN
3          1.374906                NaN               NaN
4         -0.044229                NaN               NaN

     mobile_subscriptions       internet_users  secure_internet_servers_total  \
0  less than 1 per person    0 per 1000 people                            NaN
1  less than 1 per person  154 per 1000 people                   2.623624e+06
2  more than 1 per person    90 per 100 people                   1.656589e+09
3  more than 1 per person    76 per 100 people                   6.625072e+08
```

```
4  more than 1 per person  350 per 1000 people                  2.832808e+07

  improved_sanitation women_parliament_seats_rate  life_expectancy
0         high access                    [0%-25%)        69.494195
1          low access                    [0%-25%)        59.237366
2             no info                     unknown        81.300000
3    very high access                   [25%-50%)        81.373197
4    very high access                   [25%-50%)        73.193561
```

[15]:
```
#Explore columns
data.columns
```

[15]:
```
Index(['surface_area', 'agricultural_land', 'forest_area',
       'armed_forces_total', 'urban_pop_major_cities',
       'urban_pop_minor_cities', 'national_income', 'inflation_annual',
       'inflation_monthly', 'inflation_weekly', 'mobile_subscriptions',
       'internet_users', 'secure_internet_servers_total',
       'improved_sanitation', 'women_parliament_seats_rate',
       'life_expectancy'],
      dtype='object')
```

[16]:
```
#Description
data.describe()
```

[16]:
```
       surface_area  agricultural_land   forest_area  armed_forces_total  \
count  3.620000e+02       3.580000e+02  3.570000e+02        3.180000e+02
mean   4.021884e+06       1.594881e+08  1.204151e+08        9.849864e+05
std    1.234491e+07       4.964143e+08  3.796623e+08        2.994686e+06
min    3.030000e+01       3.000000e+02  0.000000e+00        5.000000e+01
25%    2.783000e+04       1.054198e+06  4.951445e+05        1.218000e+04
50%    2.037745e+05       5.360256e+06  3.928535e+06        5.352500e+04
75%    1.081610e+06       4.221935e+07  2.241297e+07        2.598000e+05
max    1.343253e+08       5.067600e+09  4.132117e+09        2.720662e+07

       urban_pop_major_cities  urban_pop_minor_cities  inflation_annual  \
count              360.000000              360.000000        146.000000
mean                27.659456               29.175242          1.681539
std                 20.512885               21.206494          0.980308
min                  0.091444                0.074575         -2.372263
25%                 10.624625               11.013743          1.202953
50%                 24.459439               26.735127          1.762683
75%                 38.587177               43.499418          2.485675
max                 92.409069               89.142904          2.997694

       inflation_monthly  inflation_weekly  secure_internet_servers_total  \
count         156.000000         20.000000                  3.520000e+02
mean            0.475969          0.396478                  2.949654e+08
std             0.153430          0.203583                  7.234006e+08
min             0.250543          0.209993                  4.002500e+04
```

```
25%              0.347799          0.232118                  3.468446e+06
50%              0.459790          0.297938                  2.671228e+07
75%              0.577340          0.537541                  2.173937e+08
max              0.810152          0.781527                  8.207343e+09
```

```
       life_expectancy
count       362.000000
mean         71.059691
std           8.332818
min          48.850634
25%          65.469854
50%          73.238024
75%          77.125610
max          83.480488
```

[17]:
```
# Write your code here
```

[19]:
```
data.dtypes
```

[19]:
```
surface_area                   float64
agricultural_land              float64
forest_area                    float64
armed_forces_total             float64
urban_pop_major_cities         float64
urban_pop_minor_cities         float64
national_income                 object
inflation_annual               float64
inflation_monthly              float64
inflation_weekly               float64
mobile_subscriptions            object
internet_users                  object
secure_internet_servers_total  float64
improved_sanitation             object
women_parliament_seats_rate     object
life_expectancy                float64
dtype: object
```

[20]:
```
data['mobile_subscriptions'].value_counts()
```

[20]:
```
more than 1 per person    188
less than 1 per person    164
unknown                     7
more than 2 per person      2
more than 3 per person      1
Name: mobile_subscriptions, dtype: int64
```

1.3 Visualization, Modeling, Machine Learning

Can you construct a reliable model that predicts the life expectancy of an area (country, region, group of countries) using socioeconomic variables and identify how different features influence

their decision? Please explain your findings effectively to technical and non-technical audiences using comments and visualizations, if appropriate. - **Build an optimized model that effectively solves the business problem. - The model would be evaluated on the basis of Mean Absolute Error. - Read the Test.csv file and prepare features for testing.**

[22]:
```
#Loading Test data
test_data=pd.read_csv('test.csv',index_col=0)
test_data.head()
```

[22]:

	surface_area	agricultural_land	forest_area	armed_forces_total	urban_pop_major_cities	urban_pop_minor_cities	national_income	inflation_annual	inflation_monthly	inflation_weekly	mobile_subscriptions	internet_users	secure_internet_servers_total	improved_sanitation	women_parliament_seats_rate
9	322460.0	2.088892e+07	1.054769e+07	NaN	0.846584	51.919416	low	2.569961	NaN	NaN	less than 1 per person	84 per 1000 people	1849926.0	very low access	[0%-25%)
16	513120.0	2.220651e+07	1.641032e+07	453550.0	42.139810	5.803190	low	2.184886	NaN	NaN	more than 1 per person	289 per 1000 people	17983312.0	very high access	[0%-25%)
19	18580.0	1.872230e+05	8.527691e+05	NaN	1.699056	67.396944	unknown	NaN	NaN	NaN	less than 1 per person	66 per 100 people	240458015.0	very high access	unknown
23	112490.0	3.252347e+06	4.857911e+06	20000.0	48.602426	4.934574	low	NaN	0.430158	NaN	less than 1 per person	178 per 1000 people	9427882.0	high access	
28	783560.0	3.911844e+07	1.171853e+07	612800.0	43.734006	28.635994	medium low	NaN	0.624424	NaN	less than 1 per person	46 per 100 people	50379814.0	very high access	

```
23                    [0%-25%)
28                    [0%-25%)
```

[ ]:
```
# Write your code here
```

The government wants to know what are the most important features for your model. Can you tell them?

Task:

- **Visualize the top 20 features and their feature importance.**

[ ]:
```
# Write your code here
```

Task:

- **Submit the predictions on the test dataset using your optimized model** For each record in the test set (`Test.csv`), you must predict the value of the `life_expectancy` variable. You should submit a CSV file with a header row and one row per test entry. The file (submissions.csv) should have exactly 2 columns:

The file (`submissions.csv`) should have exactly 2 columns: - **id** - **life_expectancy**

[ ]:
```
# Write your code here
```

[ ]:
```
#Submission
submission_df.to_csv('submissions.csv',index=False)
```