Northeastern University

ASSIGNMENT FRONT SHEET

Course Name: **ALY6040 Data Mining Applications**

Professor Name: **Nagadeepa Shanmuganathan**

Student Name: **Dong Quoc Tuong (Lukas)**

Student Class: **Fall 2019 CPS** **Term: Winter 2021**

Module 3: Data Mangling and Data Wrangling & Logistic Regression, Decision Trees, and Random Forests

Completion Date: February 7th **Due Time:12:00am**

Statement of Authorship

I confirm that this work is my own. Additionally, I confirm that no part of this coursework, except where clearly quoted and referenced, has been copied from material belonging to any other person e.g. from a book, handout, another student. I am aware that it is a breach of Northeastern University's regulations to copy the work of another without clear acknowledgement and that attempting to do so renders me liable to disciplinary procedures. To this effect, I have uploaded my work onto Turnitin and have ensured that I have made any relevant corrections to my work prior to submission.

☒ **Tick here** to confirm that your paper version is identical to the version submitted through Turnitin

The dataset we are working on called Gapminder.org, detailing the values for life expectancy, GDP per capita, and population, and every five years from 1952 -2007 for approximately 150 countries. In this assignment, we will be using 2 packages: dplyr and tidyr

1/ dplyr

dplyr is a grammar of data manipulation, giving the coders a set of verbs that allow them to solve the most common data manipulation problems ("A Grammar of Data Manipulation • dplyr," 2020)

First, we have a look at "filter" in dplyr. It allows you to only iinlcude information that satisfy the condition you set up in the beginning. For example, you can see that we only include any inputs that has "lifeExp" < 29 or "country"= "Mexico" or "Mexico, Afghanistan"

```
> filter(gapminder, lifeExp < 29)
# A tibble: 2 x 6
  country     continent  year lifeExp     pop gdpPercap
  <fct>       <fct>     <int>   <dbl>   <int>     <dbl>
1 Afghanistan Asia       1952    28.8 8425333      779.
2 Rwanda      Africa     1992    23.6 7290203      737.
> filter(gapminder, country == "Mexico")
# A tibble: 12 x 6
   country continent  year lifeExp       pop gdpPercap
   <fct>   <fct>     <int>   <dbl>     <int>     <dbl>
 1 Mexico  Americas   1952    50.8  30144317     3478.
 2 Mexico  Americas   1957    55.2  35015548     4132.
 3 Mexico  Americas   1962    58.3  41121485     4582.
 4 Mexico  Americas   1967    60.1  47995559     5755.
 5 Mexico  Americas   1972    62.4  55984294     6809.
 6 Mexico  Americas   1977    65.0  63759976     7675.
 7 Mexico  Americas   1982    67.4  71640904     9611.
 8 Mexico  Americas   1987    69.5  80122492     8688.
 9 Mexico  Americas   1992    71.5  88111030     9472.
10 Mexico  Americas   1997    73.7  95895146     9767.
11 Mexico  Americas   2002    74.9 102479927    10742.
12 Mexico  Americas   2007    76.2 108700891    11978.
> filter(gapminder, country %in% c("Mexico", "Afghanistan"))
# A tibble: 24 x 6
   country     continent  year lifeExp      pop gdpPercap
   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
 1 Afghanistan Asia       1952    28.8  8425333      779.
 2 Afghanistan Asia       1957    30.3  9240934      821.
 3 Afghanistan Asia       1962    32.0 10267083      853.
 4 Afghanistan Asia       1967    34.0 11537966      836.
 5 Afghanistan Asia       1972    36.1 13079460      740.
```

There are two other ways do it. The first method is to identify the column inside the dataset with "$" and the second is using "subset" while retaining the same code structure like "filter"

```
gapminder[gapminder$lifeExp < 29, ]          ## repea
j] indexing is distracting
 A tibble: 2 x 6
 country     continent  year lifeExp     pop gdpPercap
 <fct>       <fct>     <int>   <dbl>   <int>     <dbl>
 Afghanistan Asia       1952    28.8 8425333      779.
 Rwanda      Africa     1992    23.6 7290203      737.
 subset(gapminder, country == "Mexico")         ## almos
... but wait ...
 A tibble: 12 x 6
  country continent  year lifeExp      pop gdpPercap
  <fct>   <fct>     <int>   <dbl>    <int>     <dbl>
1 Mexico  Americas   1952    50.8 30144317     3478.
2 Mexico  Americas   1957    55.2 35015548     4132.
3 Mexico  Americas   1962    58.3 41121485     4582.
4 Mexico  Americas   1967    60.1 47995559     5755.
5 Mexico  Americas   1972    62.4 55984294     6809.
6 Mexico  Americas   1977    65.0 63759976     7675.
```

Second, we start to dig a little bit deeper in Pipe " %>% " Operator. This operator forwards a value or a result from the expression into the next function call/ expression. Consequently, removing duplication and make your code looks cleaner, readable and more efficient to the readers. For example, codes like this

gapminder %>% head

will have the same result like this

head(gapminder)

```
gapminder %>% head # this...
A tibble: 6 x 6
country     continent  year lifeExp      pop gdpPercap
<fct>       <fct>     <int>   <dbl>    <int>     <dbl>
Afghanistan Asia       1952    28.8  8425333      779.
Afghanistan Asia       1957    30.3  9240934      821.
Afghanistan Asia       1962    32.0 10267083      853.
Afghanistan Asia       1967    34.0 11537966      836.
Afghanistan Asia       1972    36.1 13079460      740.
Afghanistan Asia       1977    38.4 14880372      786.
head(gapminder) # ...is the same as this!
A tibble: 6 x 6
country     continent  year lifeExp      pop gdpPercap
<fct>       <fct>     <int>   <dbl>    <int>     <dbl>
Afghanistan Asia       1952    28.8  8425333      779.
Afghanistan Asia       1957    30.3  9240934      821.
Afghanistan Asia       1962    32.0 10267083      853.
Afghanistan Asia       1967    34.0 11537966      836.
Afghanistan Asia       1972    36.1 13079460      740.
Afghanistan Asia       1977    38.4 14880372      786.

gapminder %>% head(3) # can pass arguments! this...
A tibble: 3 x 6
country     continent  year lifeExp      pop gdpPercap
<fct>       <fct>     <int>   <dbl>    <int>     <dbl>
Afghanistan Asia       1952    28.8  8425333      779.
Afghanistan Asia       1957    30.3  9240934      821.
Afghanistan Asia       1962    32.0 10267083      853.
head(gapminder, 3) # ...is the same as this!
A tibble: 3 x 6
country     continent  year lifeExp      pop gdpPercap
<fct>       <fct>     <int>   <dbl>    <int>     <dbl>
Afghanistan Asia       1952    28.8  8425333      779.
Afghanistan Asia       1957    30.3  9240934      821.
Afghanistan Asia       1962    32.0 10267083      853.
```

```
> select(gapminder, year, lifeExp) # this...
# A tibble: 1,704 x 2
    year lifeExp
   <int>   <dbl>
 1  1952    28.8
 2  1957    30.3
 3  1962    32.0
 4  1967    34.0
 5  1972    36.1
 6  1977    38.4
 7  1982    39.9
 8  1987    40.8
 9  1992    41.7
10  1997    41.8
# ... with 1,694 more rows
> gapminder %>% select(year, lifeExp) # ...is the same as this!
# A tibble: 1,704 x 2
    year lifeExp
   <int>   <dbl>
 1  1952    28.8
 2  1957    30.3
 3  1962    32.0
 4  1967    34.0
 5  1972    36.1
 6  1977    38.4
 7  1982    39.9
 8  1987    40.8
 9  1992    41.7
10  1997    41.8
```

“%>%” is so much better when typing out because it does not have to use the dplyr package.

```
> gapminder %>%
+   select(year, lifeExp) %>%
+   head(4) # doesn't have to be a dplyr function
# A tibble: 4 x 2
   year lifeExp
  <int>   <dbl>
1  1952    28.8
2  1957    30.3
3  1962    32.0
4  1967    34.0
```

Compared to the base R, “%>%” allows coders to fix the code much easier because they can tell exactly the issue of the code and sometimes do not need to type in one consecutive line.

```
gapminder %>%
  filter(country == "Cambodia") %>%
  select(-continent, -lifeExp) # same as
cap)
A tibble: 12 x 4
 country   year      pop gdpPercap
 <fct>    <int>    <int>     <dbl>
 Cambodia  1952  4693836      368.
 Cambodia  1957  5322536      434.
 Cambodia  1962  6083619      497.
 Cambodia  1967  6960067      523.
 Cambodia  1972  7450606      422.
 Cambodia  1977  6978607      525.
 Cambodia  1982  7272485      624.
 Cambodia  1987  8371791      684.
```

Third, the “mutate” adds a new column to the data frame that you are working. For example, f we decide to filter out to only any inputs of Cambodia, then select all the variable except “continent” , “lifeExp” and mutate/ create a new column gdp = pop * gdpPercap, we will have something like this

```
gapminder %>%
  filter(country == "Cambodia") %>%
  select(-continent, -lifeExp) %>%
  mutate(gdp = pop * gdpPercap)
A tibble: 12 x 5
  country   year     pop gdpPercap          gdp
  <fct>    <int>   <int>     <dbl>        <dbl>
1 Cambodia  1952 4693836      368.  1729534398.
2 Cambodia  1957 5322536      434.  2310184671.
3 Cambodia  1962 6083619      497.  3023033308.
4 Cambodia  1967 6960067      523.  3643123977.
5 Cambodia  1972 7450606      422.  3141354496.
6 Cambodia  1977 6978607      525.  3663574552.
7 Cambodia  1982 7272485      624.  4541488550.
```

Forth, the “summarise” or “summarize” adds new column when grouping to together. For In the first example, we see that we calculate the “mean_gdp” for Cambodia because we use filter option but for the second, we calculate for all of them using the additional “group_by” function

```
gapminder %>%
  filter(country == "Cambodia") %>%
  select(-continent, -lifeExp) %>%
  mutate(gdp = pop * gdpPercap) %>%
  group_by(country) %>%
  summarize(mean_gdp = mean(gdp)) %>%
  ungroup() # if you use group_by, also use
later
A tibble: 1 x 2
 country     mean_gdp
 <fct>          <dbl>
Cambodia 6596612377.
## summarize for all countries (replaces ou
gapminder %>%
  select(-continent, -lifeExp) %>%
  mutate(gdp = pop * gdpPercap) %>%
  group_by(country) %>%
  summarize(mean_gdp = mean(gdp)) %>%
  ungroup() # if you use group_by, also use
later
A tibble: 142 x 2
  country           mean_gdp
  <fct>                <dbl>
1 Afghanistan  12709647583.
2 Albania       9094669267.
3 Algeria      96735171261.
4 Angola       25532681843.
5 Argentina   266754123835.
6 Australia   320253755823.
7 Austria     158579002935.
8 Bahrain       7694793798.
9 Bangladesh   80648494456.
0 Belgium     197371599665.
```

[2/ tidyr]

While dplyr is certainly helpful, researchers also employ tidyr as well to create tidy dataset. Tidy data is a normalized data that is stored in a standard way and can be used to analyze or visualize insights.

First, using “gather”, we can group the different columns together into one. “gather” is often used with “separate”, which turns a single character column into multiple columns. Thus, there are four ways to write the code as followed

```r
## practice tidyr::gather() wide to long
gap_long <- gap_wide %>%
  gather(key   = obstype_year,
         value = obs_values,
         -continent, -country)
# or
gap_long <- gap_wide %>%
  gather(key   = obstype_year,
         value = obs_values,
         dplyr::starts_with('pop'),
         dplyr::starts_with('lifeExp'),
         dplyr::starts_with('gdpPercap'))
# or (but always be wary of numerics because they co
gap_long <- gap_wide %>%
  gather(key   = obstype_year,
         value = obs_values,
         3:38)  # could also do -1, -2: 'not column

## gather() and separate() to create our original ga
gap_long <- gap_wide %>%
  gather(key   = obstype_year,
         value = obs_values,
         -continent, -country) %>%
  separate(obstype_year,
           into = c('obs_type','year'),
           sep="_")
```

And the result turns the unnormalized dataset in the left to the noramlized one in the right, which is much cleaner for process

	continent	country	gdpPercap_1952	gdpPercap_1957	gdpPercap_1962	gdp
1	Africa	Algeria	2449.0082	3013.9760	2550.8169	32
2	Africa	Angola	3520.6103	3827.9405	4269.2767	55
3	Africa	Benin	1062.7522	959.6011	949.4991	10
4	Africa	Botswana	851.2411	918.2325	983.6540	12
5	Africa	Burkina Faso	543.2552	617.1835	722.5120	79
6	Africa	Burundi	339.2965	379.5646	355.2032	41
7	Africa	Cameroon	1172.6677	1313.0481	1399.6074	15
8	Africa	Central African Republic	1071.3107	1190.8443	1193.0688	11
9	Africa	Chad	1178.6659	1308.4956	1389.8176	11
10	Africa	Comoros	1102.9909	1211.1485	1406.6483	18
11	Africa	Congo Dem. Rep.	780.5423	905.8602	896.3146	86
12	Africa	Congo Rep.	2125.6214	2315.0566	2464.7832	26

=>

	continent	country	obstype_year	obs_values
1	Africa	Algeria	gdpPercap_1952	2449.0082
2	Africa	Angola	gdpPercap_1952	3520.6103
3	Africa	Benin	gdpPercap_1952	1062.7522
4	Africa	Botswana	gdpPercap_1952	851.2411
5	Africa	Burkina Faso	gdpPercap_1952	543.2552
6	Africa	Burundi	gdpPercap_1952	339.2965
7	Africa	Cameroon	gdpPercap_1952	1172.6677
8	Africa	Central African Republic	gdpPercap_1952	1071.3107
9	Africa	Chad	gdpPercap_1952	1178.6659
10	Africa	Comoros	gdpPercap_1952	1102.9909
11	Africa	Congo Dem. Rep.	gdpPercap_1952	780.5423
12	Africa	Congo Rep.	gdpPercap_1952	2125.6214
13	Africa	Cote d'Ivoire	gdpPercap_1952	1388.5947

Second, “spread” is the opposite of “gather”, spreading the data from one column into multiple ones. Here are three ways to do it

```r
## spread() from normal to wide
gap_normal <- gap_long %>%
  spread(obs_type, obs_values) %>%
  select(country, continent, year, lifeExp, pop, gdpPercap)
# or
gap_normal <- gap_long %>%
  spread(obs_type, obs_values)
gap_normal <- gap_normal[,names(gapminder)]
```

And the result will turns the normalized dataset in the left to the one in the right

	continent	country	obs_type	year	obs_values
1	Africa	Algeria	gdpPercap	1952	2449.0082
2	Africa	Angola	gdpPercap	1952	3520.6103
3	Africa	Benin	gdpPercap	1952	1062.7522
4	Africa	Botswana	gdpPercap	1952	851.2411
5	Africa	Burkina Faso	gdpPercap	1952	543.2552
6	Africa	Burundi	gdpPercap	1952	339.2965
7	Africa	Cameroon	gdpPercap	1952	1172.6677
8	Africa	Central African Republic	gdpPercap	1952	1071.3107
9	Africa	Chad	gdpPercap	1952	1178.6659
10	Africa	Comoros	gdpPercap	1952	1102.9909
11	Africa	Congo Dem. Rep.	gdpPercap	1952	780.5423

=>

	country	continent	year	lifeExp	pop	gdpPercap
1	Algeria	Africa	1952	43.077	9279525	2449.0082
2	Algeria	Africa	1957	45.685	10270856	3013.9760
3	Algeria	Africa	1962	48.303	11000948	2550.8169
4	Algeria	Africa	1967	51.407	12760499	3246.9918
5	Algeria	Africa	1972	54.518	14760787	4182.6638
6	Algeria	Africa	1977	58.014	17152804	4910.4168
7	Algeria	Africa	1982	61.368	20033753	5745.1602
8	Algeria	Africa	1987	65.799	23254956	5681.3585
9	Algeria	Africa	1992	67.744	26298373	5023.2166
10	Algeria	Africa	1997	69.152	29072015	4797.2951
11	Algeria	Africa	2002	70.994	31287142	5288.0404

Third, "all.equal" is used to compare R objects x and y testing "near equality". In the case that they are significantly different, it will make an effort to compare them and produce a report of differences. After comparing the "gap_normal" and "gapreminder" we have this

```
 all.equal(gap_normal,gapminder)
[1] "Attributes: < Component "class": Lengths (1, 3) differ (string
e on first 1) >"
[2] "Attributes: < Component "class": 1 string mismatch >"
[3] "Component "country": Modes: character, numeric"
[4] "Component "country": Attributes: < target is NULL, current is '
>"
[5] "Component "country": target is character, current is factor"
[6] "Component "continent": Modes: character, numeric"
[7] "Component "continent": Attributes: < target is NULL, current is
>"
[8] "Component "continent": target is character, current is factor"
[9] "Component "year": Modes: character, numeric"
10] "Component "year": target is character, current is numeric"
11] "Component "lifeExp": Mean relative difference: 0.203822"
12] "Component "pop": Mean relative difference: 1.634504"
13] "Component "gdpPercap": Mean relative difference: 1.162302"
```

Forth, "unite" function allows us to paste together multiple columns into one.

Final project’s proposal

In America, there are 11,000 new cases of invasive cervical cancers being found annually. Despite that fact that the number of new cases are on the downward trend over the past decade, it still kills about 4,000 women in America and 300,000 women globally.(Fontham et al., 2020) The sooner one discovers it, the more chances they have to survive. Thus, in order to effectively eliminate such diseases, we need to create a Machine Learning algorithm that can detect the cancer as soon as possible

For the final project, I am planning to analyze the dataset: Cervical cancer (Risk Factors) from UCI. The dataset was collected at 'Hospital Universitario de Caracas' in Caracas, Venezuela. Cervical cancer has 858 inputs with variables ranging from demographic information, habits to historic medical records. Some variables have missing data as patients refused to answer due to privacy concerns. I am planning to do a mix of EDA, Clustering and Classification

https://archive.ics.uci.edu/ml/datasets/Cervical+cancer+%28Risk+Factors%29

References

A Grammar of Data Manipulation • dplyr. (2020). Retrieved February 7, 2021, from https://dplyr.tidyverse.org/

Fontham, E. T. H., Wolf, A. M. D., Church, T. R., Etzioni, R., Flowers, C. R., Herzig, A., … Smith, R. A. (2020). Cervical cancer screening for individuals at average risk: 2020 guideline update from the American Cancer Society. *CA: A Cancer Journal for Clinicians, 70*(5), 321–346. https://doi.org/10.3322/caac.21628

Tidy Messy Data • tidyr. (2020). Retrieved February 7, 2021, from https://tidyr.tidyverse.org/