Data Features

JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

Introduction

Given a dataset of M variables and N dataset points (number of samples), a feature is one of the **independent** variables in a dataset and also called as a **predictor**. Generally the input to a machine learning program is a column of a tabular dataset where each row (of N rows) is a dataset point in M dimensional space.

In our Jupyter notebooks we will use the matrix $X_{N\times M}$ as the dataset symbol without the dependent variable (also called the label, category, class, predicted variable) and we will store the dependent variable in the vector $y_{N\times 1}$.

X is a matrix. It's rows are data points, and it's columns are the features. All classifiers in `scikit-learn` can understand this data format which is based on numpy 2-dimensional arrays. Thus, for each input data point we have $x \in \mathbb{R}^M$, and we have N data points to be used in our ML pipelines. We also have $y \in \mathbb{Z}^+$ in general as the category. As an example, if we have K categories, then $y \in \{k : 0 \leq k < K, k \in \mathbb{Z}^+\}$

Example: In the following X matrix we have 3 features and 5 data points, i.e. $M = 3$ and $N = 5$. We also have 5 values for dependent variable y.

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \\ x_{51} & x_{52} & x_{53} \end{bmatrix}, y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix}$$

Such that in practice,

$$X = \begin{bmatrix} 4.9 & 3. & 1.4 \\ 4.7 & 3.2 & 1.3 \\ 4.6 & 3.1 & 1.5 \\ 5. & 3.6 & 1.4 \\ 5.4 & 3.9 & 1.7 \end{bmatrix}, y = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 2 \end{bmatrix}$$

where the dependent variable y has levels from the alphabet $\Sigma = \{0, 1, 2\}$, i.e. there are 3 categories in the given dataset.

The predicted variable or label is the **dependent** variable, and it is *dependent* on the **independent variables** or features. This amount of dependence can be sometimes high and sometimes very low depending on the dataset or the nature of the problem (again, dataset expresses this). If there is no correlation (or fully independent), then the dataset may not be suitable for the problem at hand and/or we may have to remove that feature from the dataset.

As an example, for numerical variables, the Pearson correlation coefficient of two variables x (lower case x, a feature) and y is defined as:

$$r_{xy} = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}}$$, where $\bar{x}$ and $\bar{y}$ are sample means.

Ideally, we need a good correlation (close to 1 or -1) between the independent and dependent (predicted) variables so our ML model would actually work.

Question: Can correlation coefficient be used for determining important features for the machine learning model?

Data types

- Numerical - Can be integer $\mathbb{Z}$ or floating point $\mathbb{R}$. Generally it is safe to convert all numerical variables to floating point variables
- Nominal - The variable values are drawn from a finite set of **levels** or from an alphabet Σ
  - Ordinal type are nominals with a natural order, such as small, medium, big
- Binary - The variable values can be either `0` or `1` (or, `False` or `True`). Some algorithms work fast on this kind of values, especially constrained optimization related methods
- String - May not be used directly unless the ML program (preprocessing) knows how to deal with it
- Date - May not be used directly unless the ML program (preprocessing) knows how to deal with it. It might be a good idea to convert (or map) dates to some integer numbers - for example, Excel handles dates in this manner
- More complicated features, e.g. a DNA sequence (sequence of {A,C,G,T} letters) - Other, simpler, features need to be extracted from the input sequence so that this higher-level feature can be used in an ML program

Nominal to Numerical Conversion

One possible way of converting nominal variables to numerical is **one-hot encoding**:

1. During preprocessing count the number of levels in the set of possible levels a nominal variable v_{nom} takes. Such as, L different levels, $k = 1, \ldots, L$.
2. Create L binary variables for that nominal variable v_{nom} where each row will have a binary zero for $L - 1$ binary variables except for the jth level which corresponds to the level-j when v_{nom} takes a value of level-j.

Following above procedure, a nominal variable with a cardinality of L results in L many binary variable creation (and dropping the original nominal variable itself). In other words, each unique level of that nominal variable is mapped to a binary variable. Note that, for the sake of this representation, storage space is wasted.

Also observe that the one-hot encoded variables are like *Cartesian basis* of linear algebra, e.g. a feature one-hot encoded to 3 dimensions of x, y, z.

Conversion of nominal variables to numerical is an important step for many numerical-only classifiers, such as neural networks, support vector machines, and linear regression.

Note that using `sklearn.preprocessing.LabelEncoder` will impose *ordinal* numerical values which may not be correct and cause bias, therefore it should be avoided. Instead, use `OneHotEncoder`.

Example One-hot Encoding

The nominal variable T take **levels** from the $\Sigma = \{\text{low}, \text{medium}, \text{high}\}$. Numerical conversion involves each unique level being mapped to one of the T_i binary vectors.

Nominal Variable `T`	`T0`	`T1`	`T2`
low	1	0	0
medium	0	1	0
low	1	0	0
high	0	0	1
high	0	0	1
low	1	0	0

Numerical to Nominal Conversion

Generally, histograms, binning and bin boundaries are used to group numerical values into levels or one-hot encoded variables.

Online Dataset Sources

The **UCI KDD** online repository has various datasets which can be used for analysis, machine learning and several application fields, such as GIS, cybersecurity, NLP, etc. The origin of some datasets go back to more than 20 years sourced from competitions, challenges, grants, etc. Researchers and students use these datasets and share their experiences using a common platform.
Source: UCI Knowledge Discovery in Databases Archive http://kdd.ics.uci.edu/

Kaggle data repository has various datasets which are used for Kaggle competitions. The website also has tools to examine the features on-site. This source is one of the largest. Go to the Kaggle dataset source: https://www.kaggle.com/datasets

KDnuggets is another web page which encompasses almost everything (posts, news, datasets, tutorials, forums, webinars, software, etc.) that is relevant to machine learning and data analysis.
Source: KDnuggets Datasets for Data Mining and
https://www.kdnuggets.com/datasets/index.html

The rest of the notebook will demonstrate three different datasets from these repositories.

1. UCI KDD archive → 1990 US Census data
2. Kaggle → Graduate Admissions data
3. Kaggle → The Human Freedom Index data

Download the data files from UCI KDD website and Kaggle website (by registering to Kaggle -using a disposable email address- if necessary).

Important Note: About physical dataset file shared among teams. Comparing machine learning models, and measuring performances for model selection is heavily dependent on the input dataset. Thus, if a comparison between models and a comparison among different experiments or teams results are at hand, then the dataset shared among teams or different set of experiments ***must*** be exactly the *same* dataset. Moreover, to ensure the validity, the exact same file should be shared among multiple teams or between different models pipelines.

WATCH THE MODULE VIDEO for downloading files from online sources.

Dataset Curation

In the following cells we use the downloaded and previously **cleaned** data files:

1. `USCensus1990.data.csv`
2. `Admission_Predict.csv`
3. `hfi_cc_2018_cleaned.csv`

Note that there are two dataset *cleaning* tasks before a machine learning model development can begin:

- Cleaning the data so the framework understands the data right, i.e. formatting, removing confusing symbols, quotes, etc.
- Cleaning (preprocessing) the data to improve the ML task, i.e. imputing values, removing outliers, removing incorrect dataset values, deriving variables, selecting

variables, etc.

Both cleaning steps are crucial in preparing the dataset for model development.

Quote: "As data scientists, our job is to extract signal from noise." (ref: KDnuggets)

Let's see what our data files contain:

In [1]:
```
%matplotlib inline
import matplotlib.pyplot as plt
plt.rcParams["figure.dpi"] = 72
import numpy as np
import pandas as pd


# Locate and load the data file
df = pd.read_csv('../../../EP_datasets/USCensus1990.data.csv')

# Sanity check
print(f'N rows={len(df)}, M columns={len(df.columns)}')
df.head()
```

```
N rows=2458285, M columns=69
```

Out[1]:

	caseid	dAge	dAncstry1	dAncstry2	iAvail	iCitizen	iClass	dDepart	iDisabl1	iDisabl2	...
0	10000	5	0	1	0	0	5	3	2	2	...
1	10001	6	1	1	0	0	7	5	2	2	...
2	10002	3	1	2	0	0	7	4	2	2	...
3	10003	4	1	2	0	0	1	3	2	2	...
4	10004	7	1	1	0	0	0	0	2	2	...

5 rows × 69 columns

In [2]:
```
# Locate and load the data file
df = pd.read_csv('../../../EP_datasets/Admission_Predict_Ver1.1.csv')

# Sanity check
print(f'N rows={len(df)}, M columns={len(df.columns)}')
df.head()
```

```
N rows=500, M columns=9
```

Out[2]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65

In [3]:
```
# Locate and load the data file
df = pd.read_csv('../../../EP_datasets/hfi_cc_2018.csv')

# Sanity check
print(f'N rows={len(df)}, M columns={len(df.columns)}')
df.head()
```

```
N rows=1458, M columns=123
```

Out[3]:

	year	ISO_code	countries	region	pf_rol_procedural	pf_rol_civil	pf_rol_criminal	pf_
0	2016	ALB	Albania	Eastern Europe	6.661503	4.547244	4.666508	5.2917
1	2016	DZA	Algeria	Middle East & North Africa	NaN	NaN	NaN	3.8195
2	2016	AGO	Angola	Sub-Saharan Africa	NaN	NaN	NaN	3.4518
3	2016	ARG	Argentina	Latin America & the Caribbean	7.098483	5.791960	4.343930	5.7447
4	2016	ARM	Armenia	Caucasus & Central Asia	NaN	NaN	NaN	5.0032

5 rows × 123 columns

The Human Freedom Index Dataset

Opening the `hfi_cc_2018` CSV data file in Weka is tricky as it needs two modifications as in below:

- Using a text editor, change the value `"d'Ivoire"` to `"dIvoire"` by removing the single quote. The single quote is used by Weka to mark nominal variables.
- Using a text editor add single quotes to the feature name **region** to mark it as nominal. Weka wants to see single quotes in the variable name (in the header) to be able to load the type of the variables correctly.

This particular example shows that data mining, machine learning frameworks such as Weka have their own standards that the model developer has to pay attention.

Weka Framework

Weka is a data analytics framework (open-source, Java based) with very strong ML and data mining abilities. The name is inspired from a bird native to New Zealand. To install:

1. Download and install **64-bit** Java JRE https://www.java.com/en/download/

2. Download Weka Linux distro zip file from https://www.cs.waikato.ac.nz/ml/weka/downloading.html

and extract the zip to `C:\weka` on your computer's local disk. Use Windows command prompt:

1. Check the version of Java: `java -version` so that make sure the `java` on the path is reflected to the one downloaded.

2. On a command prompt, run `java -Xmx8g -jar c:\weka\weka.jar` (8 GB heap space to be used for big data files - make sure your computer supports, or adjust this value)

WATCH THE RELATED VIDEO for opening, using Weka, preprocessing and running Random Forest classifier.

Graduate Admissions Dataset

Let's open the data file `Admission_Predict.csv` in Weka. Click on `Explorer` and open the CSV file with `Open File` button. We need a dependent variable to predict or do something with it. Let's pick `A9 - Chance of Admit` (`A_XX` is the attribute which starts from index 1). We need to convert this variable to a categorical variable.

Steps to preprocess:

1. In Filter, `AddExpression -E "ifelse (A9 > 0.9, 1, 0)" -N Admit` then press `Apply`
2. In Filter, `NumericToNominal -R last`
3. In Filter, `RenameNominalValues -R last -N "0:No, 1:Yes"`

After preprocessing pick the RandomForest (RF) classifier from `Classifier-Trees-RandomForest`. Run it with 10-fold cross validation, with `Start` button. Observe the outcome.

Question: Why do you think RF model performance results 100%?

Now remove the variable `"Serial No."` (Why useless?) and remove `"Chance of Admit"` variable (`Remove` button down below). Remember we categorized it to the variable named `"Admit"`.

Question: Why do you think RF model performance is less than 100% now?

Feature Exploration

Download the data file `Suicide Rates Overview 1985 to 2016` from Kaggle website (by registering to Kaggle -using a disposable email address- if necessary) and let's explore it.

A data and feature exploration is crucial for any machine learning model at hand. Ideally the dataset feature exploration is conducted with a subject-matter expert (SME) next to the data scientist.

A feature exploration helps to understand the features, their relevance to the problem, and the dependent variable. Often independent feature and dependent variable relations are visible and machine learning algorithms find these patterns to build learning models.

In [4]:
```
# Visualizations
import seaborn as sns; sns.set(style="ticks", color_codes=True)

# Locate and load the data file
dfOrg = pd.read_csv('../../../EP_datasets/suicide-rates-overview-1985-to-2016/m

# Sanity
print(f'#rows={len(dfOrg)} #columns={len(dfOrg.columns)}')
dfOrg.head()
```

```
#rows=27820 #columns=12
```

Out[4]:

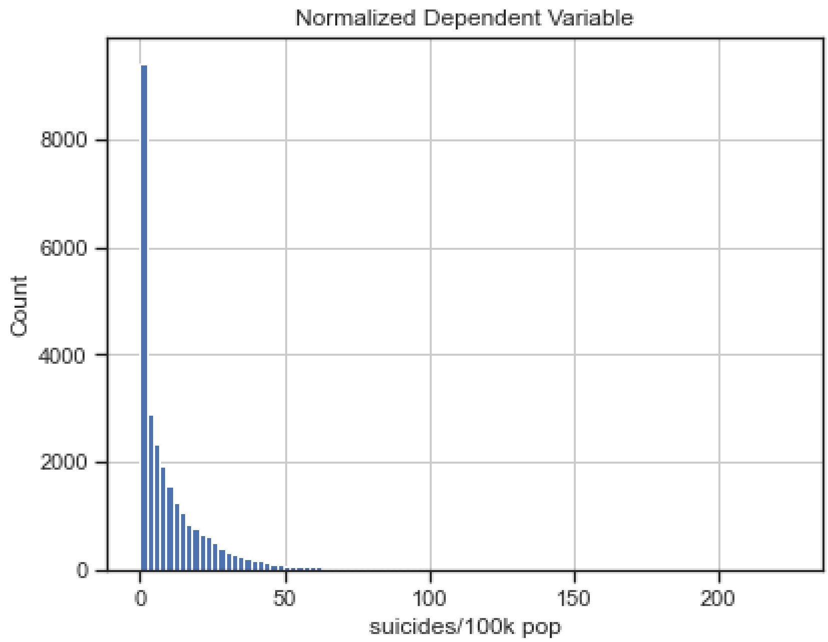
	country	year	sex	age	suicides_no	population	suicides/100k pop	country-year	HDI for year	gdp_
0	Albania	1987	male	15-24 years	21	312900	6.71	Albania1987	NaN	215
1	Albania	1987	male	35-54 years	16	308000	5.19	Albania1987	NaN	215
2	Albania	1987	female	15-24 years	14	289700	4.83	Albania1987	NaN	215
3	Albania	1987	male	75+ years	1	21800	4.59	Albania1987	NaN	215
4	Albania	1987	male	25-34 years	9	274300	3.28	Albania1987	NaN	215

In [5]:
```
# Check unique values in generation
dfOrg['generation'].unique()
```

Out[5]:
```
array(['Generation X', 'Silent', 'G.I. Generation', 'Boomers',
       'Millenials', 'Generation Z'], dtype=object)
```

In [6]:
```
# Plot the dependent variable
dfOrg['suicides/100k pop'].hist(bins=100)
plt.title('Normalized Dependent Variable')
```

```
plt.xlabel('suicides/100k pop')
plt.ylabel('Count')
plt.show()
```



In [7]:

```
# Check types of features
dfOrg.dtypes
```

Out[7]:

```
country                 object
year                     int64
sex                     object
age                     object
suicides_no              int64
population               int64
suicides/100k pop      float64
country-year            object
HDI for year           float64
 gdp_for_year ($)        int64
gdp_per_capita ($)       int64
generation              object
dtype: object
```

In [8]:

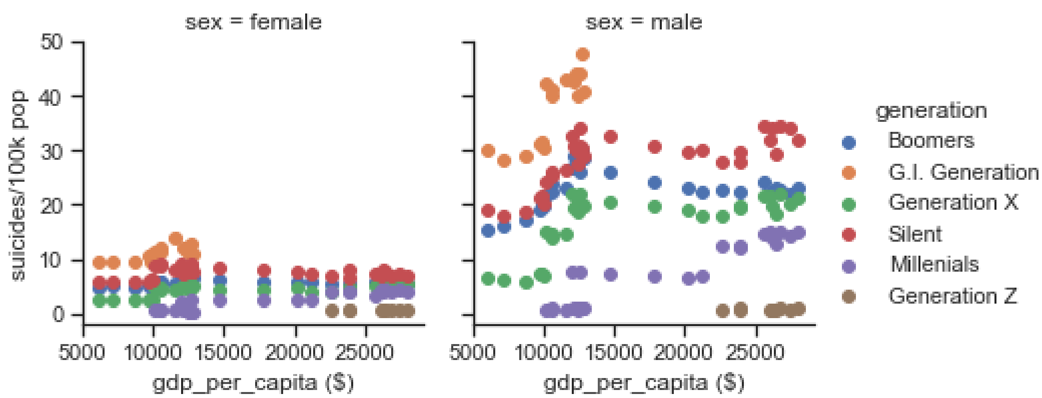
```
# Aggregate over sex, year and generation
df2 = dfOrg.copy()
df2 = df2.groupby(['sex', 'year', 'generation']).mean(numeric_only=True)
df2 = df2.reset_index()
print(len(df2))

# Sanity
df2.head()
```

```
292
```

Out[8]:

	sex	year	generation	suicides_no	population	suicides/100k pop	HDI for year	gdp_for_year ($)
0	female	1985	Boomers	101.958333	1.802462e+06	4.740208	0.699162	1.926471e+11
1	female	1985	G.I. Generation	136.125000	1.121183e+06	9.500208	0.699162	1.926471e+11
2	female	1985	Generation X	52.510417	2.030158e+06	2.544479	0.699162	1.926471e+11
3	female	1985	Silent	197.416667	2.582628e+06	5.831875	0.699162	1.926471e+11
4	female	1986	Boomers	105.375000	1.836482e+06	5.090208	NaN	2.302251e+11

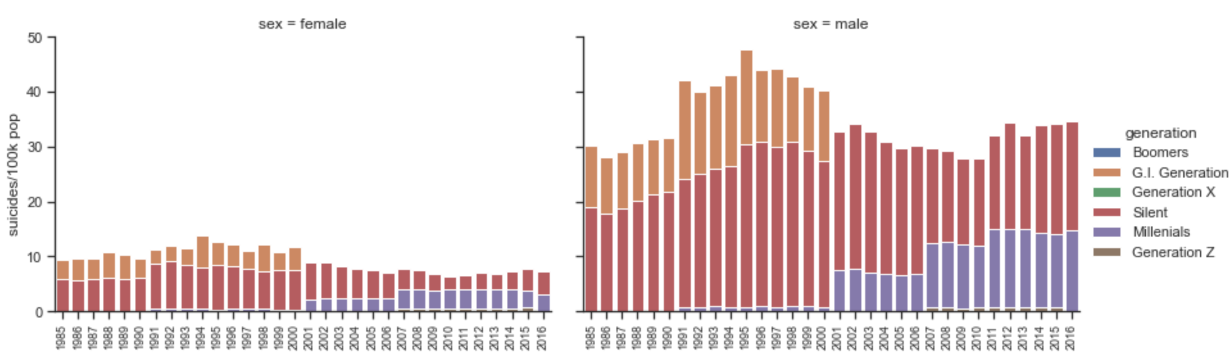
In [9]:
```
# Plot for data exploration
g = sns.FacetGrid(df2, col='sex', hue='generation')
g.map(plt.scatter, 'gdp_per_capita ($)', 'suicides/100k pop')
g.add_legend();
```



In [10]:
```
# Check the year range for the plot x-axis order
print(df2['year'].min(), df2['year'].max())
```

```
1985 2016
```

In [11]:
```
# Plot for data exploration
g = sns.FacetGrid(df2, col='sex', hue='generation', height=4, aspect=1.5)
g.map(sns.barplot, 'year', 'suicides/100k pop', order=np.arange(1985,2017))
g.set_xticklabels(rotation=90, fontsize=9)
g.add_legend();
```



In [12]:
```
# Aggregate over age and year
df3 = dfOrg.copy()
df3 = df3.groupby(['age', 'year']).mean(numeric_only=True)
df3 = df3.reset_index()
```

```
print(len(df3))

# Sanity
df3.head()
```

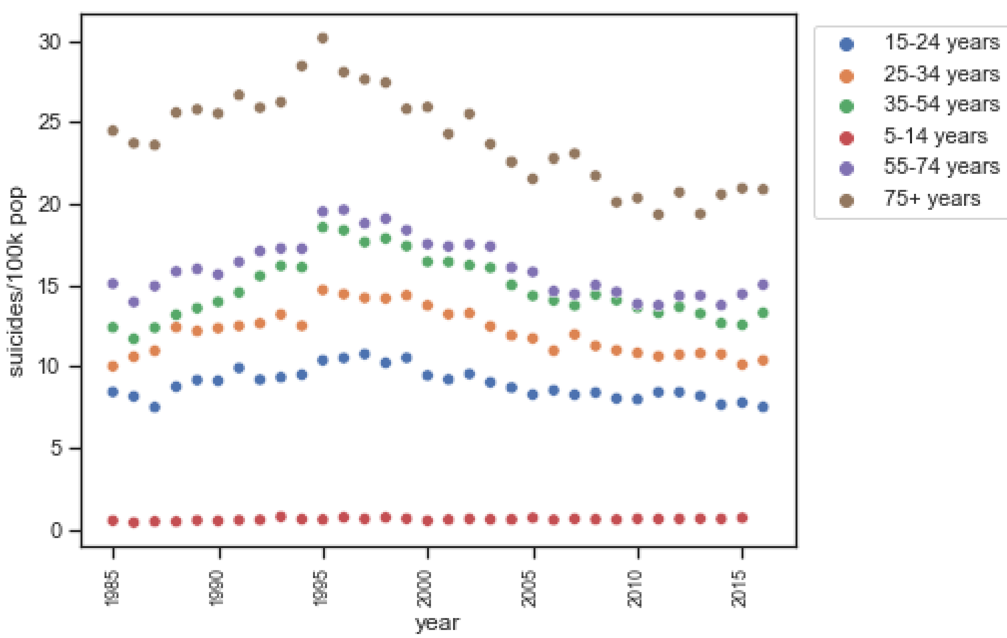
191

Out[12]:

	age	year	suicides_no	population	suicides/100k pop	HDI for year	gdp_for_year ($)	gdp_per_cap
0	15-24 years	1985	186.145833	2.051817e+06	8.429688	0.699162	1.926471e+11	6091.229
1	15-24 years	1986	188.156250	2.075402e+06	8.152083	NaN	2.302251e+11	7126.104
2	15-24 years	1987	152.148148	1.931372e+06	7.487870	NaN	2.403856e+11	8712.592
3	15-24 years	1988	156.500000	2.000362e+06	8.750918	NaN	2.985675e+11	9983.857
4	15-24 years	1989	179.192308	2.114683e+06	9.160481	NaN	3.070805e+11	9725.038

In [13]:

```
# More plots
ax = sns.scatterplot(x='year', y='suicides/100k pop', hue='age', data=df3)
plt.xticks(rotation=90, fontsize=9)
plt.legend(bbox_to_anchor=(1.01, 1.0));
```



In [14]:

```
# Aggregate over generation and country
df4 = dfOrg.copy()
df4 = df4.groupby(['generation', 'country']).mean(numeric_only=True)
df4 = df4.reset_index()
print(len(df4))
```

```
# Sanity
df4.head()
```

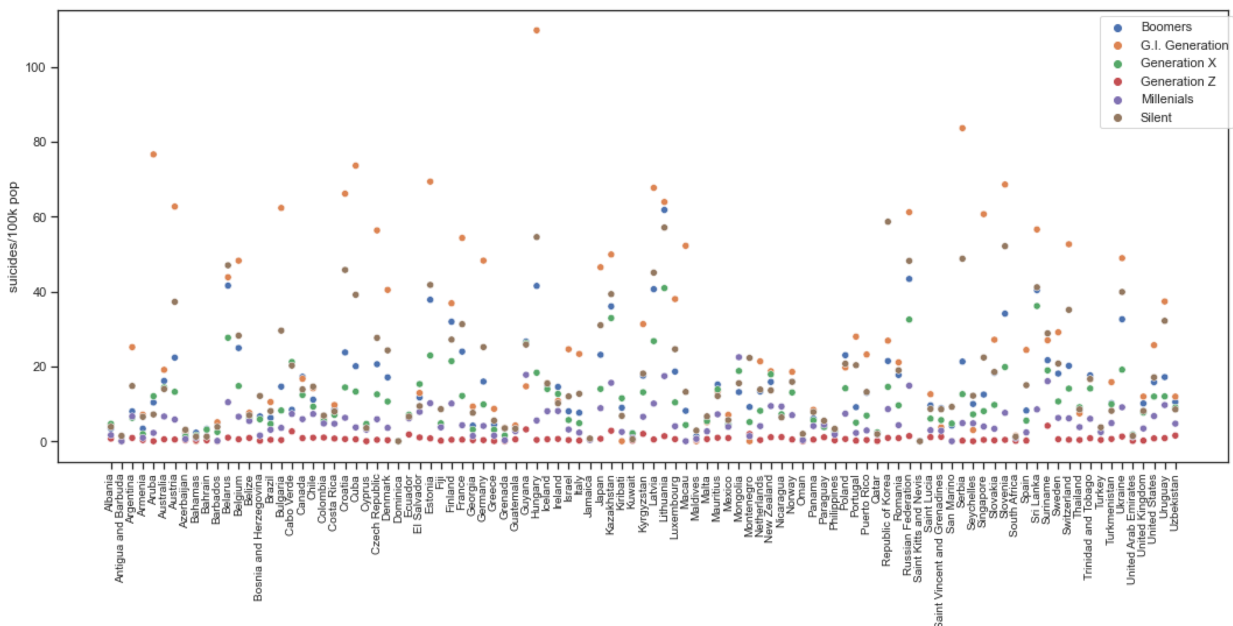
590

Out[14]:

	generation	country	year	suicides_no	population	suicides/100k pop	HDI for year	g
0	Boomers	Albania	1998.000000	12.020833	3.444468e+05	3.377708	0.656667	4.4
1	Boomers	Antigua and Barbuda	1998.580645	0.112903	8.430129e+03	1.238548	0.781667	7.6
2	Boomers	Argentina	1998.823529	278.750000	3.452533e+06	7.960147	0.776111	2.
3	Boomers	Armenia	2001.142857	11.803571	3.776771e+05	3.339286	0.685714	4.7
4	Boomers	Aruba	2003.846154	1.500000	1.546704e+04	10.336154	NaN	2.1

In [15]:

```
# More plots
plt.figure(figsize=(18, 7), dpi=72)
sns.scatterplot(x='country', y='suicides/100k pop', hue='generation', data=df4)
plt.xticks(rotation=90, fontsize=10)
plt.legend(bbox_to_anchor=(1.01, 1.0));
```



References

1. Raschka, Sebastian. Python Machine Learning Ed. 3. Packt Publishing, 2019.
2. Bojer, Casper Solheim, and Jens Peder Meldgaard. "Kaggle forecasting competitions: An overlooked learning opportunity." International Journal of Forecasting 37.2 (2021): 587-603.

Exercises

Exercise 1. Find a popular image dataset in Kaggle and report its name.

Exercise 2. In Weka try three more classifiers and report their performance. Check the dependent variable constraints in API and pick one classifier that uses numerical feature as its dependent variable.

Exercise 3. Using the plots above, answer: What should be the independent variable? Find out which age group has a stronger relation to a high suicide rate. Which gender has it?

In [16]:

```
%%html
<style>
    table {margin-left: 0 !important;}
</style>
<!-- Display markdown tables left oriented in this notebook. -->
```
