Subtle Bugs Everywhere: Generating Documentation for Data Wrangling Code

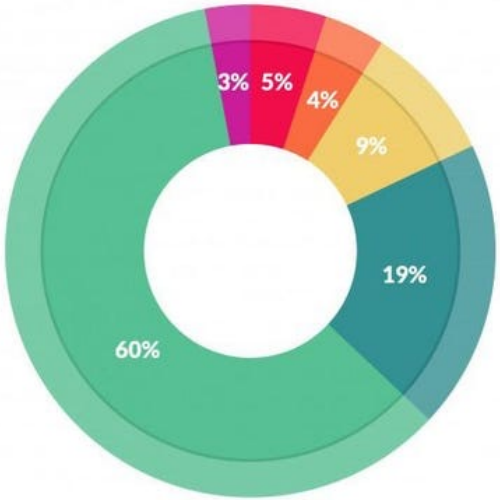
Chenyang Yang, Shurui Zhou, Jin L.C. Guo, Christian Kästner

UNIVERSITY OF TORONTO

McGill

Carnegie Mellon University

What do data scientists spend the most time doing?



What data scientists spend the most time doing

- *Building training sets: 3%*
- *Cleaning and organizing data: 60%*

Messy and disorganized data is the number one obstacle holding data scientists back. Cleaning and organizing data is the most time consuming and least interesting part of data scientists' jobs, cited by two-thirds of respondents. This

[1] Crowdflower, 2016 Data Science Report

What do data scientists spend the most time doing?

Data Wrangling: Data Cleaning + Feature Engineering

Data Science Code in Notebooks

[2]:
```
data = pd.read_csv('./data.csv')
```

Read data

[3]:
```
# first change 'Varies with device' to nan
def to_nan(item):
    if item.Size == 'Varies with device':
        return np.nan
    else:
        return item.Size

data['Size'] = data.apply(to_nan, axis=1)

# convert Size
num = data.Size.replace(r'[kM]+', '', regex=True).astype(float)
factor = data.Size.str.extract(r'[\d\.]+([KM]+)', expand=False)
factor = factor.replace(['k','M'], [10**3, 10**6]).fillna(1)
data['Size'] = num*factor.astype(int)


# fill nan
data['Size'].fillna(data['Size'].mean(),inplace = True)
```

Data wrangling

[ ]:
```
# some training code reading combined
targets = data['Target']
data.drop('Target', inplace=True, axis=1)

clf = RandomForestClassifier(n_estimators=50, max_features='sqrt')
clf = clf.fit(data, targets)
```

Learning

Data Wrangling

```
[3]: # first change 'Varies with device' to nan
def to_nan(item):
    if item.Size == 'Varies with device':
        return np.nan
    else:
        return item.Size

data['Size'] = data.apply(to_nan, axis=1)
```

```
# convert Size
num = data.Size.replace(r'[kM]+', '', regex=True).astype(float)
factor = data.Size.str.extract(r'[\d\.]+([KM]+)', expand=False)
factor = factor.replace(['k','M'], [10**3, 10**6]).fillna(1)
data['Size'] = num*factor.astype(int)
```

```
# fill nan
data['Size'].fillna(data['Size'].mean(),inplace = True)
```

```
[6]: data['Size']
```

```
[6]: 0                    19M
     1                    14M
     2                   8.7M
     3                    25M
     4                   2.8M
                  ...
     9355                2.6M
     9356                 53M
     9357                3.6M
     9358    Varies with device
     9359                 19M
     Name: Size, Length: 9360, dtype: object
```

```
[8]: data['Size']
```

```
[8]: 0       1.900000e+07
     1       1.400000e+07
     2       8.700000e+06
     3       2.500000e+07
     4       2.800000e+06
                 ...
     9355    2.600000e+06
     9356    5.300000e+07
     9357    3.600000e+06
     9358    2.294835e+07
     9359    1.900000e+07
     Name: Size, Length: 9360, dtype: float64
```

Hard to Understand

```
[3]: # first change 'Varies with device' to nan
def to_nan(item):
    if item.Size == 'Varies with device':
        return np.nan
    else:
        return item.Size

data['Size'] = data.apply(to_nan, axis=1)

# convert Size
num = data.Size.replace(r'[kM]+', '', regex=True).astype(float)
factor = data.Size.str.extract(r'[\d\.]+([KM]+)', expand=False)
factor = factor.replace(['k','M'], [10**3, 10**6]).fillna(1)
data['Size'] = num*factor.astype(int)


# fill nan
data['Size'].fillna(data['Size'].mean(),inplace = True)
```

```
[6]: data['Size']
```

```
[6]: 0                     19M
     1                     14M
     2                    8.7M
     3                     25M
     4                    2.8M
                 ...
     9355                 2.6M
     9356                  53M
     9357                 3.6M
     9358    Varies with device
     9359                  19M
     Name: Size, Length: 9360, dtype: object
```

Missing!

```
[8]: data['Size']
```

```
[8]: 0       1.900000e+07
     1       1.400000e+07
     2       8.700000e+06
     3       2.500000e+07
     4       2.800000e+06
                 ...
     9355    2.600000e+06
     9356    5.300000e+07
     9357    3.600000e+06
     9358    2.294835e+07
     9359    1.900000e+07
     Name: Size, Length: 9360, dtype: float64
```

Resulting in Subtle Bugs…

```
# convert Size
num = data.Size.replace(r'[kM]+', '', regex=True).astype(float)
factor = data.Size.str.extract(r'[\d\.]+([KM]+)', expand=False)
factor = factor.replace(['k','M'], [10**3, 10**6]).fillna(1)
data['Size'] = num*factor.astype(int)
```

Should be lowercase *k*

```
[6]: data['Size']
```

```
[6]: 0                     19M
     1                     14M
     2                    8.7M
     3                     25M
     4                    2.8M
                  ...
     9355                 2.6M
     9356                  53M
     9357                 3.6M
     9358    Varies with device
     9359                  19M
     Name: Size, Length: 9360, dtype: object
```

```
[8]: data['Size']
```

```
[8]: 0       1.900000e+07
     1       1.400000e+07
     2       8.700000e+06
     3       2.500000e+07
     4       2.800000e+06
                 ...
     9355    2.600000e+06
     9356    5.300000e+07
     9357    3.600000e+06
     9358    2.294835e+07
     9359    1.900000e+07
     Name: Size, Length: 9360, dtype: float64
```

Resulting in Subtle Bugs…

```
[32]: data['Size'][55:].head(10)
```

```
[32]: 55     54M
      56    201k
      57    3.6M
      58    5.7M
      59     17M
      60    8.6M
      61    2.4M
      62     27M
      63    2.7M
      64    2.5M
      Name: Size, dtype: object
```

```
[34]: data['Size'][55:].head(10)
```

```
[34]: 55    54000000.0
      56        201.0
      57     3600000.0
      58     5700000.0
      59    17000000.0
      60     8600000.0
      61     2400000.0
      62    27000000.0
      63     2700000.0
      64     2500000.0
      Name: Size, dtype: float64
```

Silent bug

```
# convert Size
num = data.Size.replace(r'[kM]+', '', regex=True).astype(float)
factor = data.Size.str.extract(r'[\d\.]+([KM]+)', expand=False)
factor = factor.replace(['k','M'], [10**3, 10**6]).fillna(1)
data['Size'] = num*factor.astype(int)
```

Should be lowercase *k*

Subtle Bugs Everywhere!

API misuse

```
[42]: df["Weight"].head()

[42]: 0    43
      1    72
      2    58
      3    82
      4    67
      Name: Weight, dtype: object

[43]:
      df["Weight"].astype(str).astype(int)

[43]: 0    43
      1    72
      2    58
      3    82
      4    67
      Name: Weight, dtype: int32

[44]: df["Weight"].head()

[44]: 0    43
      1    72
      2    58
      3    82
      4    67
      Name: Weight, dtype: object
```

Type unchanged!

Subtle Bugs Everywhere!

Typos

```
[3]: # first change 'Varies with device' to nan
def to_nan(item):
    if item.Size == 'Varies with device':
        return np.nan
    else:
        return item.Size

data['Size'] = data.apply(to_nan, axis=1)

# convert Size
num = data.Size.replace(r'[kM]+', '', regex=True).astype(float)
factor = data.Size.str.extract(r'[\d\.]+([KM]+)', expand=False)
factor = factor.replace(['k','M'], [10**3, 10**6]).fillna(1)
data['Size'] = num*factor.astype(int)


# fill nan
data['Size'].fillna(data['Size'].mean(),inplace = True)
```

```
[48]:
df2['Reviews_count']= df1['Reviews'].apply(lambda x: int(x))
```

Read from wrong source

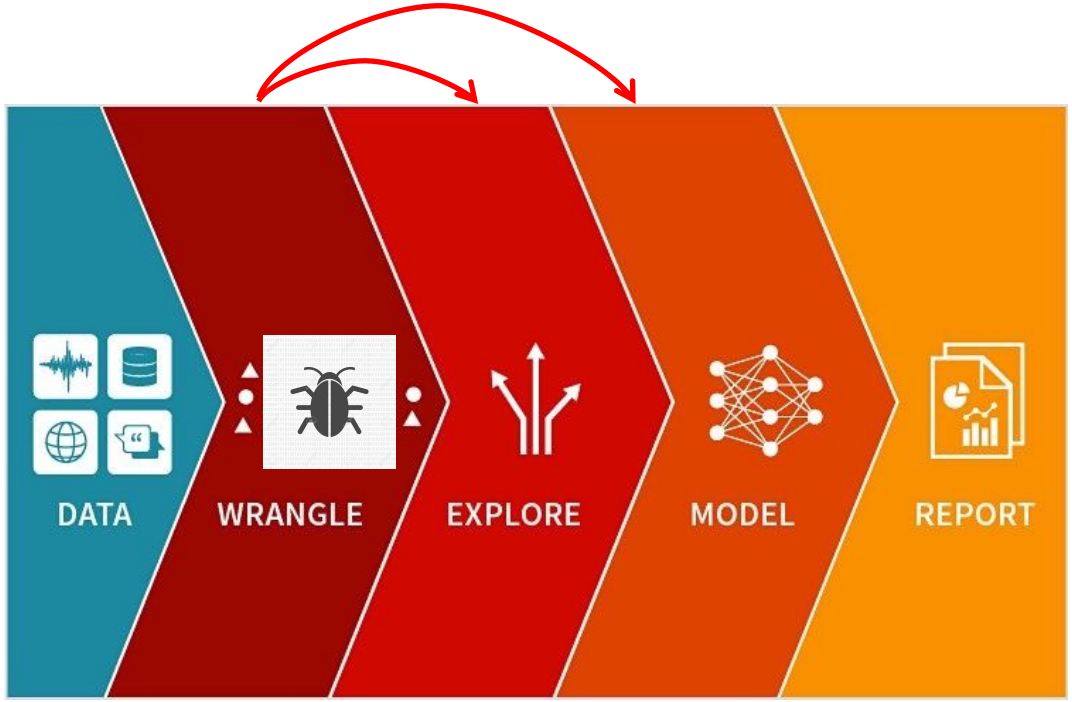
Subtle Bugs Everywhere!

Forget corner case!
3.4k -> 3.4000

Modeling

```
[ ]:
df["Release Clause"]= df["Release Clause"].replace(regex=['k'], value='000')
df["Release Clause"]= df["Release Clause"].astype(str).astype(float)
```

Subtle Bugs Everywhere!



Subtle Bugs

↓

Reduced Model Quality

[1] https://www.upgrad.com/blog/a-beginners-guide-to-data-science-and-its-applications/

Existing Tool Support is Limited

Unsuited for understanding data wrangling code!

Documentation to the Rescue

```
[3]: # first change 'Varies with device' to nan
def to_nan(item):
    if item.Size == 'Varies with device':
        return np.nan
    else:
        return item.Size

data['Size'] = data.apply(to_nan, axis=1)

# convert Size
num = data.Size.replace(r'[kM]+', '', regex=True).astype(float)
factor = data.Size.str.extract(r'[\d\.]+([KM]+)', expand=False)
factor = factor.replace(['k','M'], [10**3, 10**6]).fillna(1)
data['Size'] = num*factor.astype(int)


# fill nan
data['Size'].fillna(data['Size'].mean(),inplace = True)
```

WrangleDoc

SUMMARY

▸ INPUTS **data**

▸ OUTPUTS **data**

data -> data

changed columns: [Size]
Size = **float**(**str_transform**(Size))

	Size *	App	Category	Rating	Reviews
type	**object→float64**	object	object	float64	object
unique	**413→413**	8190	33	39	5990
range	**→[8.5, 100000000.0]**			[1.0, 5.0]	
7460	**19M→19000000.0**	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159
1637	**Varies with device→ 22948351.235594977**	Floor Plan Creator	ART_AND_DESIGN	4.1	36639
263	**201k→201.0**	Restart Navigator	AUTO_AND_VEHICLES	4	1403
	23k→23.0	Plugin:AOT v5.0	BUSINESS	3.1	4034

Identifying Relevant Variables

data

↓

```
[3]: # first change 'Varies with device' to nan
def to_nan(item):
    if item.Size == 'Varies with device':
        return np.nan
    else:
        return item.Size

data['Size'] = data.apply(to_nan, axis=1)

# convert Size
num = data.Size.replace(r'[kM]+', '', regex=True).astype(float)
factor = data.Size.str.extract(r'[\d\.]+([KM]+)', expand=False)
factor = factor.replace(['k','M'], [10**3, 10**6]).fillna(1)
data['Size'] = num*factor.astype(int)


# fill nan
data['Size'].fillna(data['Size'].mean(),inplace = True)
```

↓

data

WrangleDoc

SUMMARY

▸ INPUTS **data**

▸ OUTPUTS **data**

data -> data

changed columns: [Size]

Size = **float**(**str_transform**(Size))

	Size *	App	Category	Rating	Reviews
type	**object→float64**	object	object	float64	object
unique	**413→413**	8190	33	39	5990
range	**→[8.5, 100000000.0]**			[1.0, 5.0]	
7460	**19M→19000000.0**	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159
1637	**Varies with device→ 22948351.235594977**	Floor Plan Creator	ART_AND_DESIGN	4.1	36639
263	**201k→201.0**	Restart Navigator	AUTO_AND_VEHICLES	4	1403
	23k→23.0	Plugin:AOT v5.0	BUSINESS	3.1	4034

Concise Transformation Summary

```
[3]: # first change 'Varies with device' to nan
def to_nan(item):
    if item.Size == 'Varies with device':
        return np.nan
    else:
        return item.Size

data['Size'] = data.apply(to_nan, axis=1)

# convert Size
num = data.Size.replace(r'[kM]+', '', regex=True).astype(float)
factor = data.Size.str.extract(r'[\d\.]+([KM]+)', expand=False)
factor = factor.replace(['k','M'], [10**3, 10**6]).fillna(1)
data['Size'] = num*factor.astype(int)


# fill nan
data['Size'].fillna(data['Size'].mean(),inplace = True)
```

data -> data

changed columns: [Size]
Size = **float**(**str_transform**(Size))

	Size *	App	Category	Rating	Reviews
type	**object→float64**	object	object	float64	object
unique	**413→413**	8190	33	39	5990
range	**→[8.5, 100000000.0]**			[1.0, 5.0]	
7460	**19M→19000000.0**	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159
1637	**Varies with device→ 22948351.235594977**	Floor Plan Creator	ART_AND_DESIGN	4.1	36639
263	**201k→201.0**	Restart Navigator	AUTO_AND_VEHICLES	4	1403
	23k→23.0	Plugin:AOT v5.0	BUSINESS	3.1	4034

Highlighting Transformed Columns

```
[3]: # first change 'Varies with device' to nan
def to_nan(item):
    if item.Size == 'Varies with device':
        return np.nan
    else:
        return item.Size

data['Size'] = data.apply(to_nan, axis=1)

# convert Size
num = data.Size.replace(r'[kM]+', '', regex=True).astype(float)
factor = data.Size.str.extract(r'[\d\.]+([KM]+)', expand=False)
factor = factor.replace(['k','M'], [10**3, 10**6]).fillna(1)
data['Size'] = num*factor.astype(int)


# fill nan
data['Size'].fillna(data['Size'].mean(),inplace = True)
```

data -> data

changed columns: [Size]
Size = **float**(**str_transform**(Size))

	Size *	App	Category	Rating	Reviews
type	**object→float64**	object	object	float64	object
unique	**413→413**	8190	33	39	5990
range	**→[8.5, 100000000.0]**			[1.0, 5.0]	
7460	**19M→19000000.0**	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159
1637	**Varies with device→ 22948351.235594977**	Floor Plan Creator	ART_AND_DESIGN	4.1	36639
263	**201k→201.0**	Restart Navigator	AUTO_AND_VEHICLES	4	1403
	23k→23.0	Plugin:AOT v5.0	BUSINESS	3.1	4034

Clustering Examples

```
[3]: # first change 'Varies with device' to nan
def to_nan(item):
    if item.Size == 'Varies with device':
        return np.nan
    else:
        return item.Size

data['Size'] = data.apply(to_nan, axis=1)

# convert Size
num = data.Size.replace(r'[kM]+', '', regex=True).astype(float)
factor = data.Size.str.extract(r'[\d\.]+([KM]+)', expand=False)
factor = factor.replace(['k','M'], [10**3, 10**6]).fillna(1)
data['Size'] = num*factor.astype(int)


# fill nan
data['Size'].fillna(data['Size'].mean(),inplace = True)
```

data -> data

changed columns: [Size]
Size = **float**(**str_transform**(Size))

	Size *	App	Category	Rating	Reviews
type	**object→float64**	object	object	float64	object
unique	**413→413**	8190	33	39	5990
range	**→[8.5, 100000000.0]**			[1.0, 5.0]	
7460	**19M→19000000.0**	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159
1637	**Varies with device→ 22948351.235594977**	Floor Plan Creator	ART_AND_DESIGN	4.1	36639
263	**201k→201.0**	Restart Navigator	AUTO_AND_VEHICLES	4	1403
	23k→23.0	Plugin:AOT v5.0	BUSINESS	3.1	4034

Check more examples

WrangleDoc

File Edit View Run Kernel Tabs Settings Help

Example.ipynb

Code Python 3

[2]:
```
data = pd.read_csv('./data.csv')
```

[3]:
```
# first change 'Varies with device' to nan
def to_nan(item):
    if item.Size == 'Varies with device':
        return np.nan
    else:
        return item.Size

data['Size'] = data.apply(to_nan, axis=1)

# convert size
num = data.Size.replace(r'[kM]+', '', regex=True).astype(float)
factor = data.Size.str.extract(r'[\d\.]+([KM]+)', expand=False)
factor = factor.replace(['k','M'], [10**3, 10**6]).fillna(1)
data['Size'] = num*factor.astype(int)


# fill nan
data['Size'].fillna(data['Size'].mean(),inplace = True)
```

[ ]:
```
# some training code reading combined
targets = data['Target']
data.drop('Target', inplace=True, axis=1)

clf = RandomForestClassifier(n_estimators=50, max_features='sqrt')
clf = clf.fit(data, targets)
```

[ ]:

Autodoc Panel

SUMMARY

▸ INPUTS **data**

▸ OUTPUTS **data**

data -> data

changed columns: [Size]

Size = **float**(**str_transform**(Size))

	Size *	App	Category	Rating	Reviews	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
type	**object→float64**	object	object	float64	object	object	object	object	object	object	object	object	object
unique	**413→413**	8190	33	39	5990	19	2	73	6	115	1299	2638	31
range	**→[8.5, 100000000.0]**			[1.0, 5.0]									
7460	**19M→19000000.0**	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	10,000+	Free	0	Everyone	Art & Design	January 7, 2018	1.0.0	4.0.3 and up
1637	**Varies with device→ 22948351.235594977**	Floor Plan Creator	ART_AND_DESIGN	4.1	36639	5,000,000+	Free	0	Everyone	Art & Design	July 14, 2018	Varies with device	2.3.3 and up
263	**201k→201.0**	Restart Navigator	AUTO_AND_VEHICLES	4	1403	100,000+	Free	0	Everyone	Auto & Vehicles	August 26, 2014	1.0.1	2.2 and up
	23k→23.0	Plugin:AOT v5.0	BUSINESS	3.1	4034	100,000+	Free	0	Everyone	Business	September 11, 2015	3.0.1.11 (Build 311)	2.2 and up

Behind Documentation

Summary Synthesis

Example Selection

WrangleDoc

SUMMARY

▸ INPUTS **data**

▸ OUTPUTS **data**

data -> data

changed columns: [Size]
Size = **float**(**str_transform**(Size))

	Size *	App	Category	Rating	Reviews
type	**object→float64**	object	object	float64	object
unique	**413→413**	8190	33	39	5990
range	**→[8.5, 100000000.0]**			[1.0, 5.0]	
7460	**19M→19000000.0**	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159
1637	**Varies with device→ 22948351.235594977**	Floor Plan Creator	ART_AND_DESIGN	4.1	36639
263	**201k→201.0**	Restart Navigator	AUTO_AND_VEHICLES	4	1403
	23k→23.0	Plugin:AOT v5.0	BUSINESS	3.1	4034

Summary Synthesis

SUMMARY

▸ INPUTS **data**

▸ OUTPUTS **data**

Summary Synthesis

[3]:
```
# first change 'Varies with device' to nan
def to_nan(item):
    if item.Size == 'Varies with device':
        return np.nan
    else:
        return item.Size

data['Size'] = data.apply(to_nan, axis=1)

# convert Size
num = data.Size.replace(r'[kM]+', '', regex=True).astype(float)
factor = data.Size.str.extract(r'[\d\.]+([KM]+)', expand=False)
factor = factor.replace(['k','M'], [10**3, 10**6]).fillna(1)
data['Size'] = num*factor.astype(int)


# fill nan
data['Size'].fillna(data['Size'].mean(),inplace = True)
```

Too concise!

Dataframe *data* was changed

Column *Size* was changed with
- values ending in ‘M’ replacing ‘M’ with 10^6
- values ending in ‘k’ replacing ‘k’ with 1
- values ‘Varies with Device’ changed to the mean
- all values are converted to float.

Too detailed!

Summary Synthesis

DSL

Input-Output pair

Synthesizer

Plausible expression

Size = **float**(**str_transform**(Size))

DSL for data wrangling

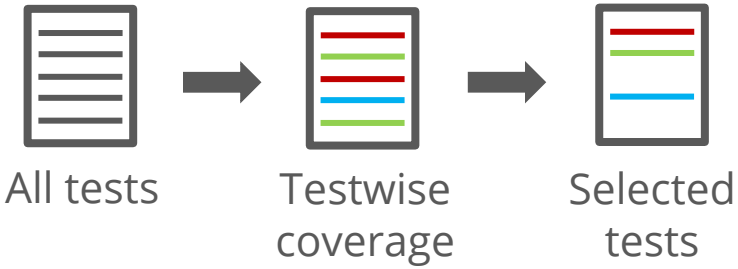
```
<col> := fillna(<col>)₂ // fill null values
       | merge(<col>)₂ // merge items to reduce cardinality
       | category(<col>)₂ // convert columns to category type
       | float(<col>)₂ | str(<col>)₂ | int(<col>)₂
       | bool(<col>)₂ | datetime64(<col>)₂ // type conversion
       | encode(<col>)₂ // encode columns in consecutive ints
       | one_hot_encoding(<col>)₂ // encode columns in 0/1 ints
       | type_convert(<col>)₃ // other type conversion
       | str_transform(<col>)₃ // unspecified string transf.
       | num_transform(<col>)₃ // unspecified numerical transf.
       | compute(<col_ref>*)₁₅ // unspecified col. computation
       | <col_ref>*₀

<col_ref> := DATAFRAME.COL₀
```

Example Selection

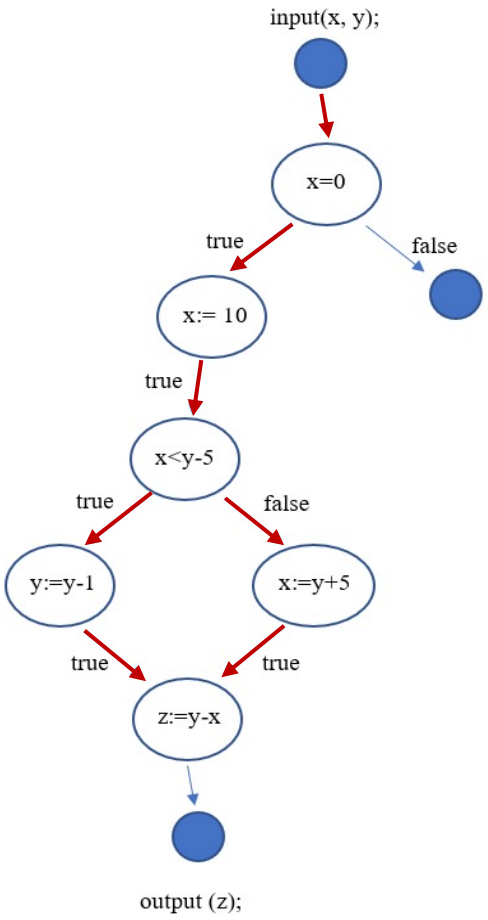
Each example is a test case

Select example → Select test case

Test Case Selection

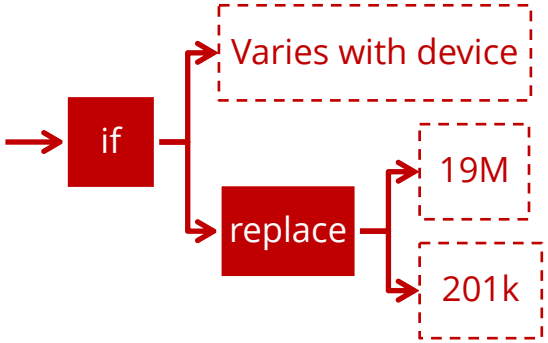


Select test case based on coverage



Example Selection

```
[3]: # first change 'Varies with device' to nan
def to_nan(item):
    if item.Size == 'Varies with device':
        return np.nan
    else:
        return item.Size

data['Size'] = data.apply(to_nan, axis=1)

# convert Size
num = data.Size.replace(r'[kM]+', '', regex=True).astype(float)
factor = data.Size.str.extract(r'[\d\.]+([KM]+)', expand=False)
factor = factor.replace(['k','M'], [10**3, 10**6]).fillna(1)
data['Size'] = num*factor.astype(int)


# fill nan
data['Size'].fillna(data['Size'].mean(),inplace = True)
```

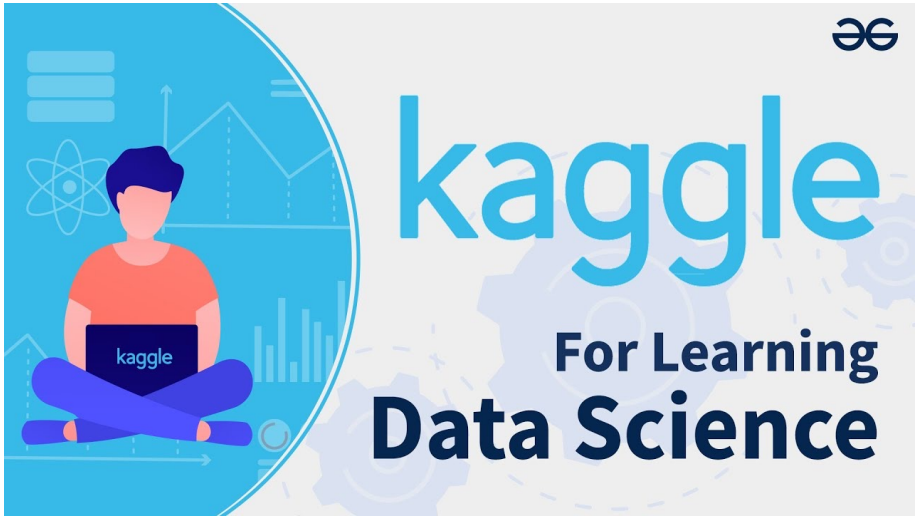


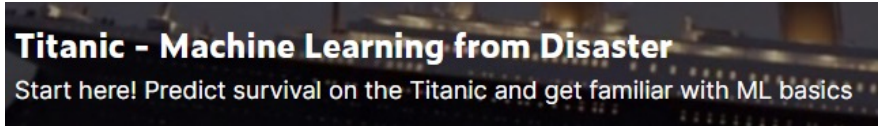
7460	19M→19000000.0
1637	Varies with device→ 22948351.235594977
263	201k→201.0

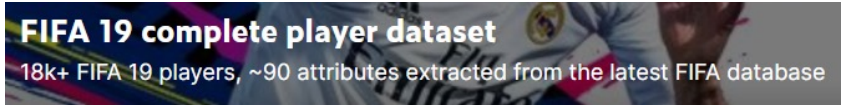
Tracking row-level branching decisions

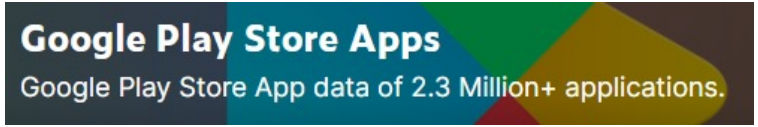
Is Our Approach Effective and Useful?

Evaluation: Effectiveness



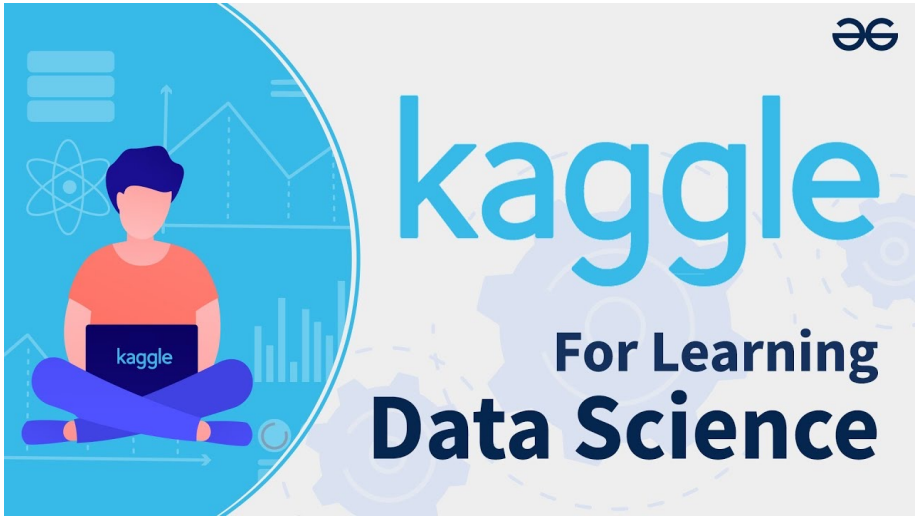






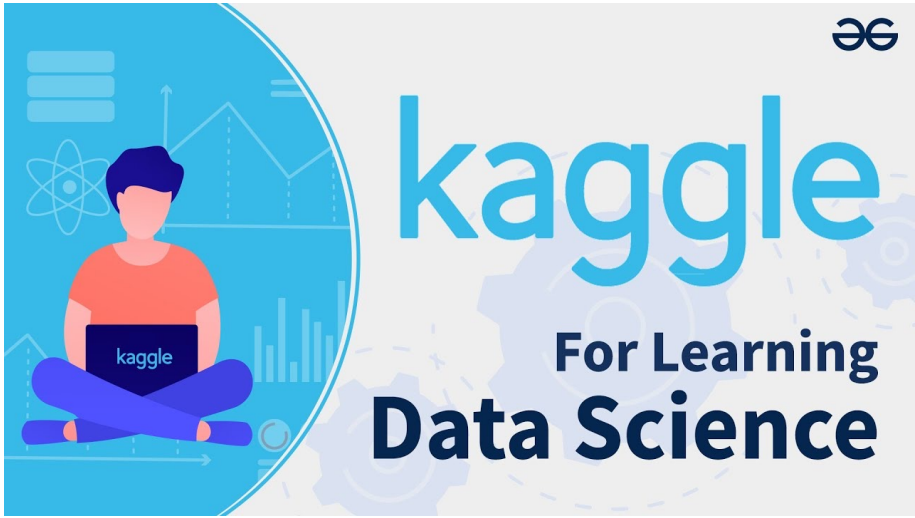
100 top notebooks from **4** datasets at **Kaggle**

Evaluation: Effectiveness



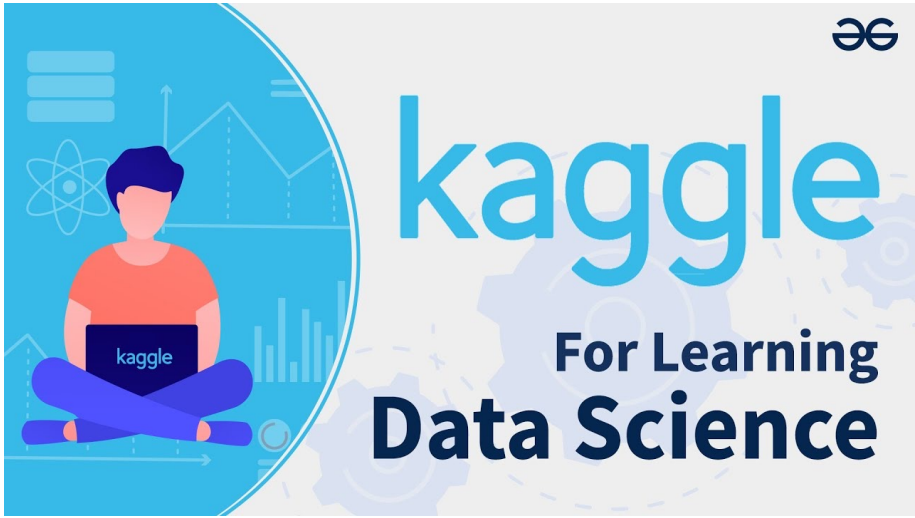
Most summaries are **correct**

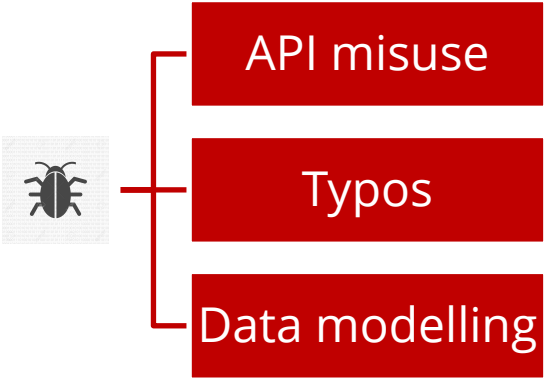
Evaluation: Effectiveness



Execution slows down by **3x** on average

Evaluation: Effectiveness





Evaluation: Usefulness

20 participants, 4 bug-finding tasks

Bugs are based on the empirical study

Evaluation: Usefulness

Users with our tool find

54% more bugs and 80% faster

Generating Documentation for Data Wrangling Code

Generating input-specific documentation to help understand data wrangling code and expose subtle wrangling bugs

- API misuse
- Typos
- Data modelling

Technical components:

Summary synthesis

Def-use analysis → Recording run-time values → **Summary synthesis**

Example selection

Def-use analysis → **Tracking branching decisions** → Clustering examples

	Size *	App	Category	Rating	Reviews
type	object→float64	object	object	float64	object
unique	413→413	8190	33	39	5990
range	→[8.5, 100000000.0]			[1.0, 5.0]	
7460	19M→19000000.0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159
1637	Varies with device→ 22948351.235594977	Floor Plan Creator	ART_AND_DESIGN	4.1	36639
263	201k→201.0	Restart Navigator	AUTO_AND_VEHICLES	4	1403
	23k→23.0	Plugin:AOT v5.0	BUSINESS	3.1	4034

Carnegie Mellon University

institute for SOFTWARE RESEARCH

https://se-phd.isri.cmu.edu/

Contact me & Read the paper! →