Spark-lean

An interactive PySpark-based Data Cleaning Library

Qintai Liu, Shuaiji Li, Yicheng Pu

Center For Data Science, NYU

Introduction

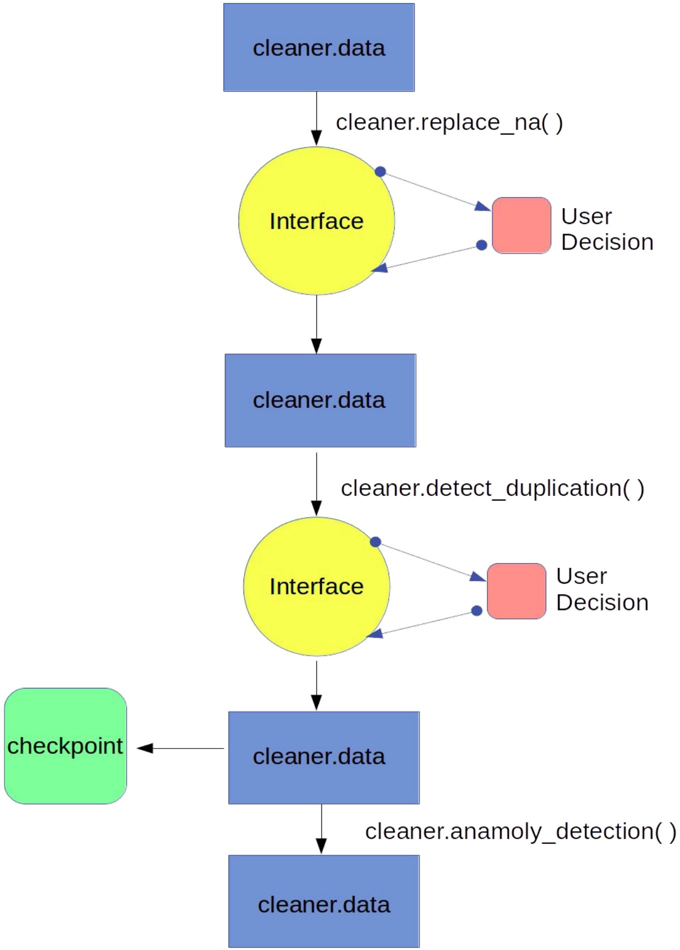
- Scalable
- Interactive
- User-friendly

Main Features:

1. Missing Value Detection
2. Numeric Features Detection
3. String Cleaning
4. Test-train Splitting
5. Similar Text Matching
6. Duplicate Column Detection
7. Useless Feature Detection
8. Anomaly detection
9. Data versioning

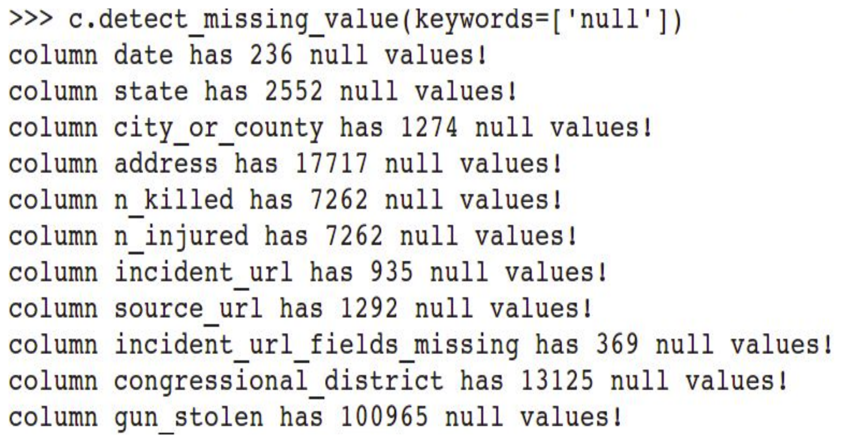
Structure

- One-class structure
- Self.data
- Self.out

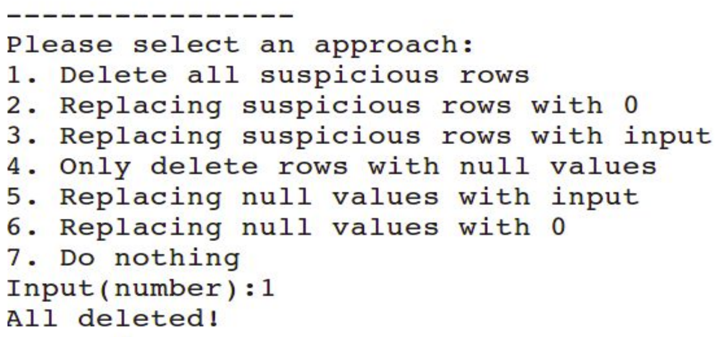


Missing Value Detection

- Null Value
- Customized Keywords
- Predefined Approach



```
>>> c.detect_missing_value(keywords=['null'])
column date has 236 null values!
column state has 2552 null values!
column city_or_county has 1274 null values!
column address has 17717 null values!
column n_killed has 7262 null values!
column n_injured has 7262 null values!
column incident_url has 935 null values!
column source_url has 1292 null values!
column incident_url_fields_missing has 369 null values!
column congressional_district has 13125 null values!
column gun_stolen has 100965 null values!
```



```
----------------
Please select an approach:
1. Delete all suspicious rows
2. Replacing suspicious rows with 0
3. Replacing suspicious rows with input
4. Only delete rows with null values
5. Replacing null values with input
6. Replacing null values with 0
7. Do nothing
Input(number):1
All deleted!
```

Numeric features detection

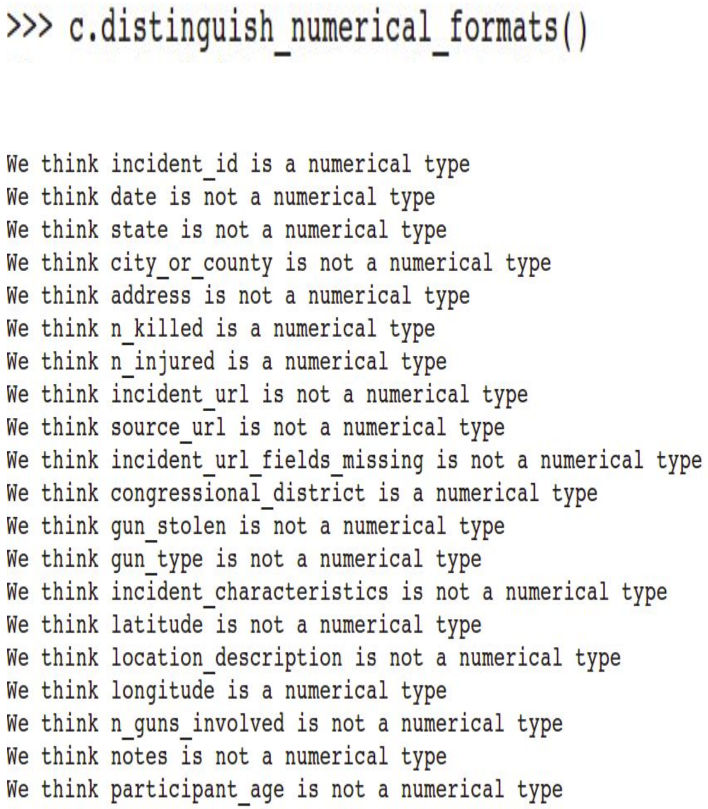
One of the default loading csv method in PySpark:

SQLContext.read.csv()

will convert every attribute to ‘string’ type

Distinguish numerical attributes:

- Pre-processing choice
- Predictive modelling

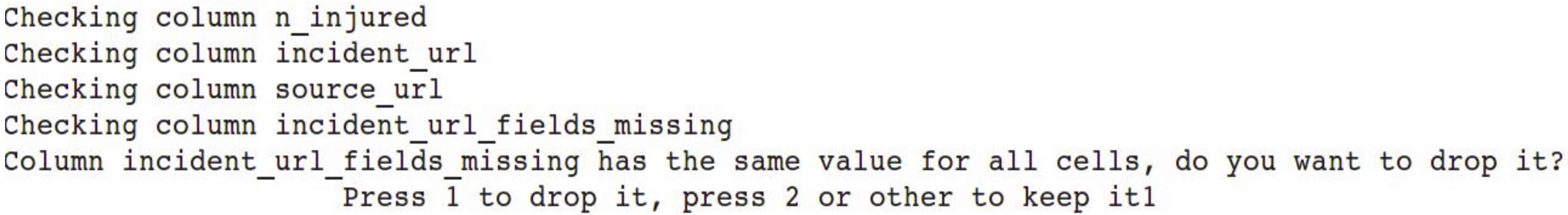


```
>>> c.distinguish_numerical_formats()

We think incident_id is a numerical type
We think date is not a numerical type
We think state is not a numerical type
We think city_or_county is not a numerical type
We think address is not a numerical type
We think n_killed is a numerical type
We think n_injured is a numerical type
We think incident_url is not a numerical type
We think source_url is not a numerical type
We think incident_url_fields_missing is not a numerical type
We think congressional_district is a numerical type
We think gun_stolen is not a numerical type
We think gun_type is not a numerical type
We think incident_characteristics is not a numerical type
We think latitude is not a numerical type
We think location_description is not a numerical type
We think longitude is a numerical type
We think n_guns_involved is not a numerical type
We think notes is not a numerical type
We think participant_age is not a numerical type
```

Useless Feature Detection

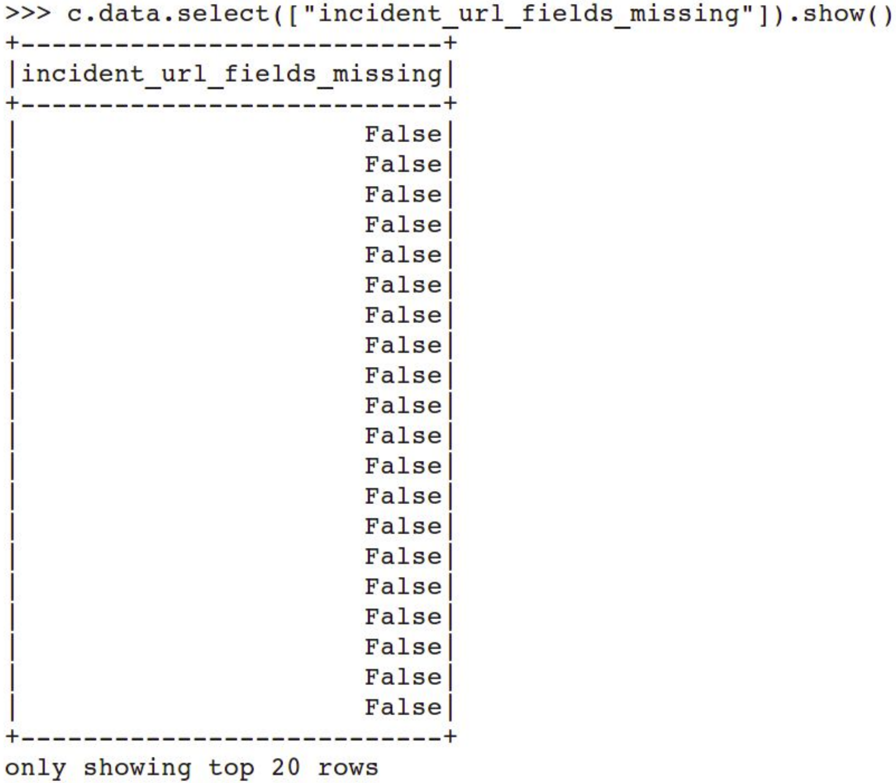
- Find the feature whose all values are all the same.
- Check the top ten values.
- If they are same:
  - Count how many values are equal to the first value of the feature.

Useless Feature Detection



```
Checking column n_injured
Checking column incident_url
Checking column source_url
Checking column incident_url_fields_missing
Column incident_url_fields_missing has the same value for all cells, do you want to drop it?
                    Press 1 to drop it, press 2 or other to keep it1
```

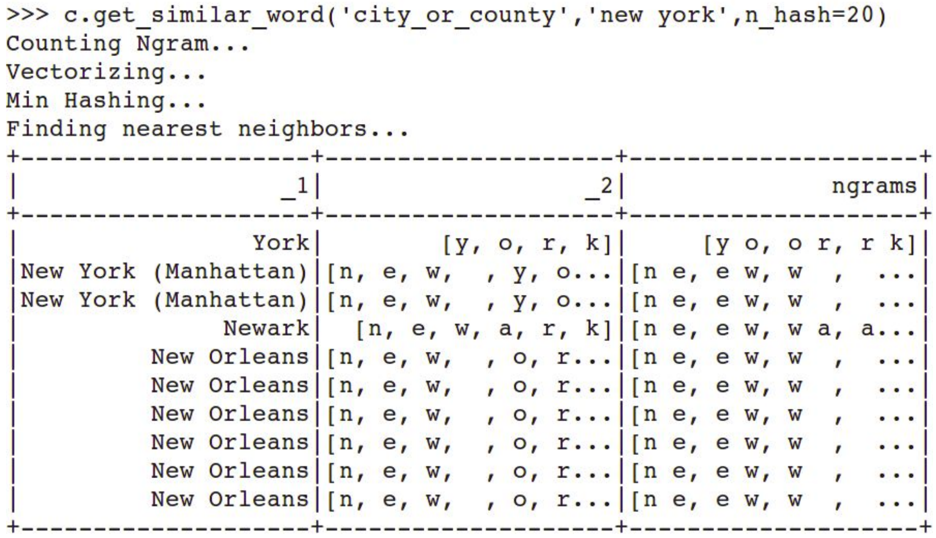
Useless Feature Detection



```
>>> c.data.select(["incident_url_fields_missing"]).show()
+---------------------------+
|incident_url_fields_missing|
+---------------------------+
|                      False|
|                      False|
|                      False|
|                      False|
|                      False|
|                      False|
|                      False|
|                      False|
|                      False|
|                      False|
|                      False|
|                      False|
|                      False|
|                      False|
|                      False|
|                      False|
|                      False|
|                      False|
|                      False|
|                      False|
+---------------------------+
only showing top 20 rows
```

Similar Text Matching

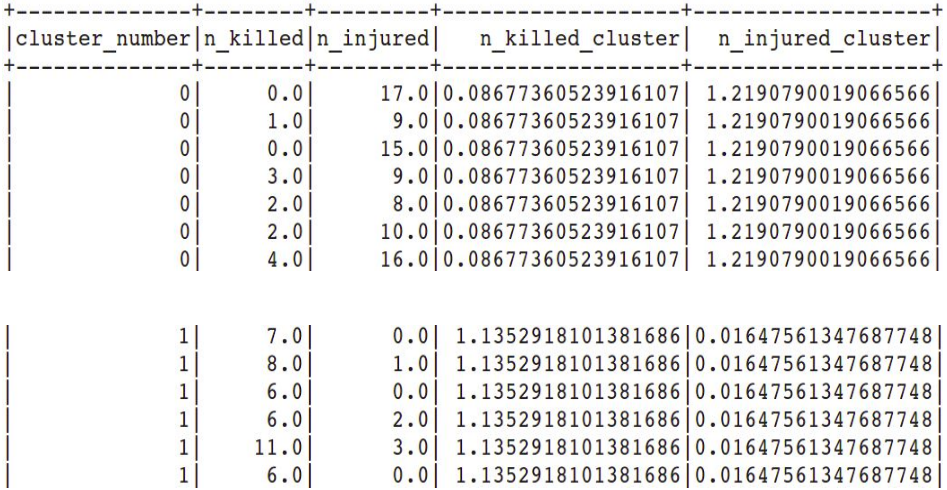
- Character Bi-gram Feature
- Min-Hash



```
>>> c.get_similar_word('city_or_county','new york',n_hash=20)
Counting Ngram...
Vectorizing...
Min Hashing...
Finding nearest neighbors...
+--------------------+--------------------+--------------------+
|                  _1|                  _2|              ngrams|
+--------------------+--------------------+--------------------+
|                York|        [y, o, r, k]|     [y o, o r, r k]|
|New York (Manhattan)|[n, e, w,  , y, o...|[n e, e w, w  ,  ...|
|New York (Manhattan)|[n, e, w,  , y, o...|[n e, e w, w  ,  ...|
|              Newark|  [n, e, w, a, r, k]|[n e, e w, w a, a...|
|         New Orleans|[n, e, w,  , o, r...|[n e, e w, w  ,  ...|
|         New Orleans|[n, e, w,  , o, r...|[n e, e w, w  ,  ...|
|         New Orleans|[n, e, w,  , o, r...|[n e, e w, w  ,  ...|
|         New Orleans|[n, e, w,  , o, r...|[n e, e w, w  ,  ...|
|         New Orleans|[n, e, w,  , o, r...|[n e, e w, w  ,  ...|
|         New Orleans|[n, e, w,  , o, r...|[n e, e w, w  ,  ...|
+--------------------+--------------------+--------------------+
```

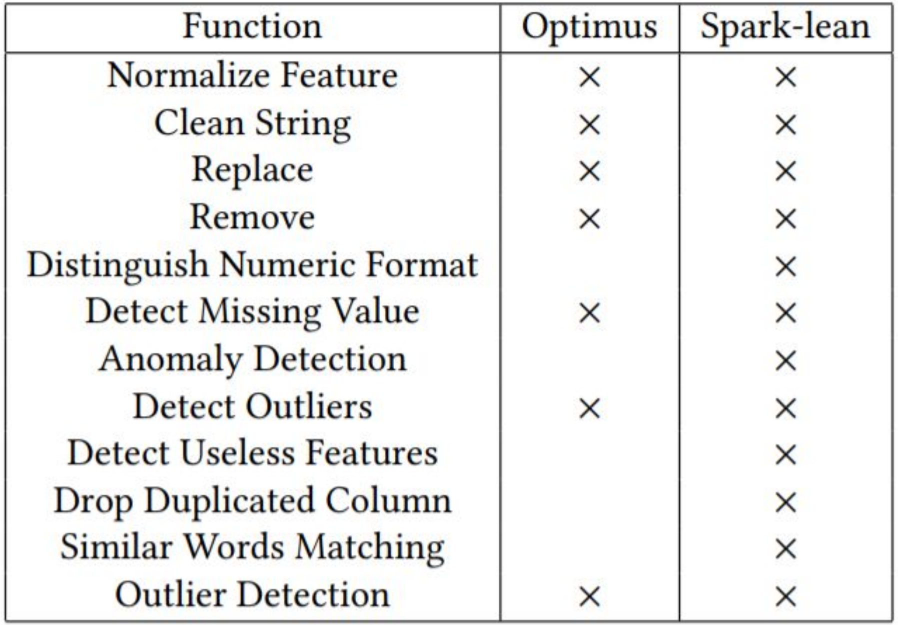
Anomaly Detection

- Standardize
- K-means Clustering
- For each cluster, calculate the mean and std of distance of all points within it to the centroid
- Find outliers by looking at their distance to their centroids (In-cluster-distance)



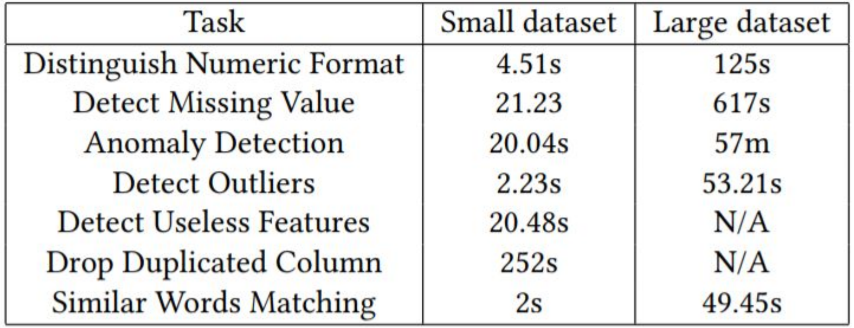
```
+--------------+--------+---------+-------------------+-------------------+
|cluster_number|n_killed|n_injured|   n_killed_cluster|  n_injured_cluster|
+--------------+--------+---------+-------------------+-------------------+
|             0|     0.0|     17.0|0.08677360523916107| 1.2190790019066566|
|             0|     1.0|      9.0|0.08677360523916107| 1.2190790019066566|
|             0|     0.0|     15.0|0.08677360523916107| 1.2190790019066566|
|             0|     3.0|      9.0|0.08677360523916107| 1.2190790019066566|
|             0|     2.0|      8.0|0.08677360523916107| 1.2190790019066566|
|             0|     2.0|     10.0|0.08677360523916107| 1.2190790019066566|
|             0|     4.0|     16.0|0.08677360523916107| 1.2190790019066566|

|             1|     7.0|      0.0| 1.1352918101381686|0.01647561347687748|
|             1|     8.0|      1.0| 1.1352918101381686|0.01647561347687748|
|             1|     6.0|      0.0| 1.1352918101381686|0.01647561347687748|
|             1|     6.0|      2.0| 1.1352918101381686|0.01647561347687748|
|             1|    11.0|      3.0| 1.1352918101381686|0.01647561347687748|
|             1|     6.0|      0.0| 1.1352918101381686|0.01647561347687748|
```

Results and Summary



Function	Optimus	Spark-lean
Normalize Feature	×	×
Clean String	×	×
Replace	×	×
Remove	×	×
Distinguish Numeric Format		×
Detect Missing Value	×	×
Anomaly Detection		×
Detect Outliers	×	×
Detect Useless Features		×
Drop Duplicated Column		×
Similar Words Matching		×
Outlier Detection	×	×

Comparison between Optimus and Spark-lean



Task	Small dataset	Large dataset
Distinguish Numeric Format	4.51s	125s
Detect Missing Value	21.23	617s
Anomaly Detection	20.04s	57m
Detect Outliers	2.23s	53.21s
Detect Useless Features	20.48s	N/A
Drop Duplicated Column	252s	N/A
Similar Words Matching	2s	49.45s

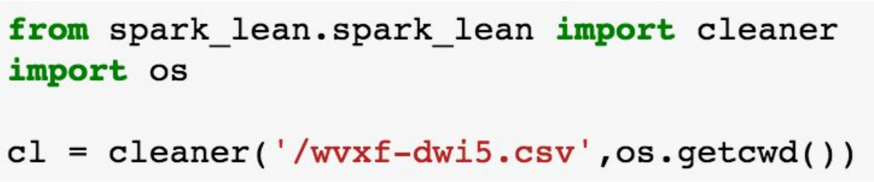
Comparison between Small Dataset(130MB) and Large Dataset(3.5GB)

Future Work

- Optimize Computational Performance
- Refine Documentation
- Support Unstructured Data
- GUI

Usage

- Make sure you have PySpark installed on your local machine
- Python 3.4+
- pip install Spark-lean



```
from spark_lean.spark_lean import cleaner
import os

cl = cleaner('/wvxf-dwi5.csv',os.getcwd())
```

Q & A