sparx Documentation

Release 0.0.1

Bastin Robin J

Aug 07, 2017

Contents

Sparx is an exclusive data preprocessing library which involves in transforming raw data into an machine understandable format. We at CleverInsight Lab took the initiative to build a better automated data preprocessing library and here it is.

CHAPTER 1

Help

TODO: write content

CHAPTER 2

Installation

Install the extension with using pip.

```
$ pip install -U sparx
```

CHAPTER 3

Quickstart

Sparx is an exclusive data preprocessing library which involves in transforming raw data into an machine understandable format. We at CleverInsight Lab took the initiative to build a better automated data preprocessing library and here it is.

Simple Usage

```
>>> from sparx.preprocess import *
```

is_categorical

Check if the given pandas series is an categorical variable `True`

```
>>> is_categorical(data[col])
>>> True
```

is_date

Return `True` if the given pandas series is an date type

```
>>> is_date(data[col])
>>> True
```

count_missing

Return the count of missing values in the given pandas.core.series

```
>>> count_missing(df['col_name'])
>>> 0
```

missing_percent

Returns the percentage of missing values in the column

```
>>> missing_percent(df['col_name'])
>>> 0
```

types

Returns the column names in groups for the given DataFrame

```
>>> types(df)
>>> {'dates': ['D'],
...  'groups': ['C', 'D'],
...  'keywords': ['C'],
...  'numbers': ['A', 'B']}
```

has_keyword

Returns `True` if any of the first 1000 non-null values in a string‘‘series‘‘ are strings that have more than `thresh` =2 separators (space, by default) in them

```
>>> has_keywords(series)
>>> False
>>> has_keywords(series, thresh=1)
>>> True
```

groupmeans

Yields the significant differences in average between every pair of groups and numbers.

```
>>> has_keywords(series)
>>> False
>>> has_keywords(series, thresh=1)
>>> True
```

describe

Return the basic description of an column in a pandas dataframe check if the column is an interger or float type

```
>>> describe(dataframe, 'Amount')
>>> {'min': 0, 'max': 100, 'mean': 50, 'median': 49 }
```

geocode

Returns `Dict` which consist of address, latitude, longitude of the given address

```
>>> geocode("172, 5th Avenue, Flatiron, Manhattan")
>>> {'latitude': 40.74111015,
...  'adress': u'172, 5th Avenue, Flatiron,
...  Manhattan, Manhattan Community Board 5, New York County, NYC,
...  New York, 10010, United States of America',
...  'longitude': -73.9903105}
```

unique_value_count

Returns the count of `unique value` fromm each column

```
>>> unique_value_count(data['name'])
>>> {'gender': {'Male': 2, 'Female': 6},
... 'age': {32: 2, 34: 2, 35: 1, 37: 1, 21: 1, 28: 1},
... 'name': {'Neeta': 1, 'vandana': 2, 'Amruta': 1, 'Vikrant': 2,
... 'vanana': 1, 'Pallavi': 1}}
```

unique_identifier

Returns a list of columns from the dataframe which consist of unique identifiers

```
>>> unique_identifier(pd.Dataframe)
>>> ['age', 'id']
```

date_split

Returns a `dictionary` of year, month, day, hour, minute and seconds

```
>>> date_split("march/1/1980")
>>> {'second': '00', 'hour': '00', 'year': '1980', 'day': '01',
... 'minute': '00', 'month': '03'}
```

dict_query_string

Returns a string which is the query formed using the given dictionary as a parameter

```
>>> query = {'name': 'Sam', 'age': 20 }

>>> dict_query_string(query)
>>> name=Same&age=20
```

encode

Returns a clean dataframe which is initially converted into *utf8* format and all categorical variables are converted into *numeric labels* also each label encoding classes are saved into a dictionary, now a tuple of first element is dataframe and second is the hash_map

```
>>> encode(pd.DataFrame())
>>> [150 rows x 6 columns], {'Species': {0: 'setosa', 1: 'versicolor', 2: 'virginica'}
↪})
```

strip_non_alphanum

Returns `List` of alphanumeric string by stripping the non alpha numeric characters

```
>>> strip_non_alphanum('epqenw49021[4;;ds..,.,uo]mfLCP'X')
>>> ['epqenw49021', '4', 'ds', 'uo', 'mfLCP', 'X']
```

word_freq_count

Returns ` dict` which consist of each words as key and its frequency count as value

```
>>> word_freq_count("hello how are you")
>>> {'a': 1, ' ': 3, 'e': 2, 'h': 2, 'l': 2, 'o': 3, 'r': 1,
... 'u': 1, 'w': 1, 'y': 1}
```

ignore_stopwords

Returns the list of words ignoring stopwords in the given list of words

```
>>> ignore_stopwords("I am basically a lazy person and i hate computers")
>>> ['I', 'basically', 'lazy', 'person', 'hate', 'computers']
```

CHAPTER 4

Authors

Development Lead

- Bastin Robins <robin@cleverinsight.co>
- Vandana Bhagat <vandana.bhagat@cleverinsight.co>
- Pallavi Murthi <pallavi.murthy@cleverinsight.co>

Contributors

- Kothandaraman Sridharan <sri@cleverinsight.co>
- Anusha Sridharan <anu.sridharan@gmail.com>
- Aruna Sridharan <aruna.sridharan@gmail.com>

CHAPTER 5

Contributing

CHAPTER 6

History

0.0.1 (pre-release on 22 July 2017)

- sparx.preprocess methods

CHAPTER 7

Indices and tables

- genindex
- modindex
- search