ydot

Release 0.0.6

Jee Vang, Ph.D.

Dec 08, 2020

CONTENTS

`ydot` is a Python API to produce PySpark dataframe models from R-like formula expressions. This project is based on patsy [pat]. As a quickstart, let's say you have a Spark dataframe with data as follows.

Table 1: Dummy Data in a Spark Dataframe

a	b	x1	x2	y
left	low	19.945536387662504	3.85214120038979	0.0
left	low	20.674308066353493	4.098585619118175	1.0
right	high	20.346647025958433	2.7107604387194626	1.0
right	mid	18.699653829045985	5.2111542692543065	1.0
left	low	21.51851187887476	2.432390426907621	1.0
right	mid	20.989823705535017	3.6774523253171734	1.0
right	high	20.277680897136328	2.4873300559969604	0.0
right	mid	19.551410645704927	2.3549674965407372	0.0
right	low	20.96196624352397	3.1665930443154995	0.0
right	mid	19.172421360793678	3.562224297579924	1.0

Now, let's say you want to model this dataset as follows.

- `y ~ x_1 + x_2 + a + b`

Then all you have to do is use the `smatrices()` function.

```
from ydot.spark import smatrices

formula = 'y ~ x1 + x2 + a + b'
y, X = smatrices(formula, sdf)
```

Observe that `y` and `X` will be Spark dataframes as specified by the formula. Here's a more interesting example where you want a model specified up to all two-way interactions.

- `y ~ (x1 + x2 + a + b)**2`

Then you could issue the code as below.

```
from ydot.spark import smatrices

formula = 'y ~ (x1 + x2 + a + b)**2'
y, X = smatrices(formula, sdf)
```

Your resulting `X` Spark dataframe will look like the following.

Table 2: Dummy Data Transformed by Formula

Intercept	a[T.right]	b[T.low]	b[T.mid]	a[T.right]:b[T.low]	a[T.right]:b[T.mid]	x1	x1:a[T.right]	x1:b[T.low]	x1:b[T.mid]	x2	x2:a[T.right]	x2:b[T.low]	x2:b[T.mid]	x1:x2
1.0	0.0	1.0	0.0	0.0	0.0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
1.0	0.0	1.0	0.0	0.0	0.0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
1.0	1.0	0.0	0.0	0.0	0.0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	55.154885818557126
1.0	1.0	0.0	1.0	0.0	1.0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
1.0	0.0	1.0	0.0	0.0	0.0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
1.0	1.0	0.0	1.0	0.0	1.0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
1.0	1.0	0.0	0.0	0.0	0.0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	50.437285161362595
1.0	1.0	0.0	1.0	0.0	1.0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
1.0	1.0	1.0	0.0	1.0	0.0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
1.0	1.0	0.0	1.0	0.0	1.0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

In general, what you get with `patsy` is what you get with `ydot`, however, there are exceptions. For example, the builtin functions such as `standardize()` and `center()` available with `patsy` will not work against Spark dataframes. Additionally, patsy allows for custom transforms, but such transforms (or user defined functions) must be visible. For now, only numpy-based transformed are allowed against continuous variables (or numeric columns).

CHAPTER

ONE

QUICKSTART

1.1 Basic

The best way to learn R-style formula syntax with ydot is to head on over to patsy [pat] and read the documentation. Below, we show very simple code to transform a Spark dataframe into two design matrices (these are also Spark dataframes), y and X, using a formula that defines a model up to two-way interactions.

```
import random
from random import choice

import numpy as np
import pandas as pd
from pyspark.sql import SparkSession

from ydot.spark import smatrices

random.seed(37)
np.random.seed(37)


def get_spark_dataframe(spark):
    n = 100
    data = {
        'a': [choice(['left', 'right']) for _ in range(n)],
        'b': [choice(['high', 'mid', 'low']) for _ in range(n)],
        'x1': np.random.normal(20, 1, n),
        'x2': np.random.normal(3, 1, n),
        'y': [choice([1.0, 0.0]) for _ in range(n)]
    }
    pdf = pd.DataFrame(data)

    sdf = spark.createDataFrame(pdf)
    return sdf


if __name__ == '__main__':
    try:
        spark = (SparkSession.builder
                 .master('local[4]')
                 .appName('local-testing-pyspark')
                 .getOrCreate())
        sdf = get_spark_dataframe(spark)

        y, X = smatrices('y ~ (x1 + x2 + a + b)**2', sdf)
```

(continues on next page)

(continued from previous page)

```
        y = y.toPandas()
        X = X.toPandas()

        print(X.head(10))
        X.head(10).to_csv('two-way-interactions.csv', index=False)
    except Exception as e:
        print(e)
    finally:
        try:
            spark.stop()
            print('closed spark')
        except Exception as e:
            print(e)
```

1.2 More

We use the code below to generate the models (data) below.

```
import random
from random import choice

import numpy as np
import pandas as pd
from pyspark.sql import SparkSession

from ydot.spark import smatrices

random.seed(37)
np.random.seed(37)


def get_spark_dataframe(spark):
    n = 100
    data = {
        'a': [choice(['left', 'right']) for _ in range(n)],
        'b': [choice(['high', 'mid', 'low']) for _ in range(n)],
        'x1': np.random.normal(20, 1, n),
        'x2': np.random.normal(3, 1, n),
        'y': [choice([1.0, 0.0]) for _ in range(n)]
    }
    pdf = pd.DataFrame(data)

    sdf = spark.createDataFrame(pdf)
    return sdf


if __name__ == '__main__':
    try:
        spark = (SparkSession.builder
                 .master('local[4]')
                 .appName('local-testing-pyspark')
                 .getOrCreate())
        sdf = get_spark_dataframe(spark)

```

(continues on next page)

(continued from previous page)

```
        formulas = [
            {
                'f': 'y ~ np.sin(x1) + np.cos(x2) + a + b',
                'o': 'transformed-continuous.csv'
            },
            {
                'f': 'y ~ x1*x2',
                'o': 'star-con-interaction.csv'
            },
            {
                'f': 'y ~ a*b',
                'o': 'star-cat-interaction.csv'
            },
            {
                'f': 'y ~ x1:x2',
                'o': 'colon-con-interaction.csv'
            },
            {
                'f': 'y ~ a:b',
                'o': 'colon-cat-interaction.csv'
            },
            {
                'f': 'y ~ (x1 + x2) / (a + b)',
                'o': 'divide-interaction.csv'
            },
            {
                'f': 'y ~ x1 + x2 + a - 1',
                'o': 'no-intercept.csv'
            }
        ]

        for item in formulas:
            f = item['f']
            o = item['o']

            y, X = smatrices(f, sdf)
            y = y.toPandas()
            X = X.toPandas()

            X.head(5).to_csv(o, index=False)

            s = f"""
            .. csv-table:: {f}
               :file: _code/{o}
               :header-rows: 1
            """
            print(s.strip())
    except Exception as e:
        print(e)
    finally:
        try:
            spark.stop()
            print('closed spark')
        except Exception as e:
            print(e)
```

You can use `numpy` functions against continuous variables.

Table 1: y ~ np.sin(x1) + np.cos(x2) + a + b

Intercept	a[T.right]	b[T.low]	b[T.mid]	np.sin(x1)	np.cos(x2)
1.0	0.0	1.0	0.0	0.8893769205406579	-0.758004200582313
1.0	0.0	1.0	0.0	0.9679261582216445	-0.5759807266894401
1.0	1.0	0.0	0.0	0.9972849995254774	-0.9086185088676886
1.0	1.0	0.0	1.0	-0.14934132364604816	0.4783416124776783
1.0	0.0	1.0	0.0	0.45523550315103734	-0.7588816501987654

The $\star$ specifies interactions and keeps lower order terms.

Table 2: y ~ x1*x2

Intercept	x1	x2	x1:x2
1.0	19.945536387662504	3.85214120038979	76.83302248278848
1.0	20.674308066353493	4.098585619118175	84.73542172597531
1.0	20.346647025958433	2.7107604387194626	55.154885818557126
1.0	18.699653829045985	5.2111542692543065	97.44678088481062
1.0	21.51851187887476	2.432390426907621	52.341422295472896

Table 3: y ~ a*b

Intercept	a[T.right]	b[T.low]	b[T.mid]	a[T.right]:b[T.low]	a[T.right]:b[T.mid]
1.0	0.0	1.0	0.0	0.0	0.0
1.0	0.0	1.0	0.0	0.0	0.0
1.0	1.0	0.0	0.0	0.0	0.0
1.0	1.0	0.0	1.0	0.0	1.0
1.0	0.0	1.0	0.0	0.0	0.0

The `:` specifies interactions and drops lower order terms.

Table 4: y ~ x1:x2

Intercept	x1:x2
1.0	76.83302248278848
1.0	84.73542172597531
1.0	55.154885818557126
1.0	97.44678088481062
1.0	52.341422295472896

Table 5: y ~ a:b

Intercept	b[T.low]	b[T.mid]	a[T.right]:b[high]	a[T.right]:b[low]	a[T.right]:b[mid]
1.0	1.0	0.0	0.0	0.0	0.0
1.0	1.0	0.0	0.0	0.0	0.0
1.0	0.0	0.0	1.0	0.0	0.0
1.0	0.0	1.0	0.0	0.0	1.0
1.0	1.0	0.0	0.0	0.0	0.0

The `/` is **quirky** according to the patsy documentation, but it is shorthand for `a / b = a + a:b`.

Table 6: y ~ (x1 + x2) / (a + b)

Intercept	x1	x2	x1:x2:a[left]	x1:x2:a[right]	x1:x2:b[T.low]	x1:x2:b[T.mid]
1.0	19.945536387662504	3.85214120038979	76.83302248278848	0.0	76.83302248278848	0.0
1.0	20.674308066353493	4.098585619118175	84.73542172597501	0.0	84.73542172597501	0.0
1.0	20.346647025958433	2.7107604387194626	0.0	55.154885818557126	0.0	0.0
1.0	18.699653829045985	5.2111542692543065	0.0	97.44678088481062	0.0	97.44678088481062
1.0	21.51851187887476	2.432390426907621	52.34142229547296	0.0	52.34142229547296	0.0

If you need to drop the `Intercept`, add `- 1` at the end. Note that one of the dummy variables for `a` is not dropped. This could be a bug with patsy.

Table 7: y ~ x1 + x2 + a - 1

a[left]	a[right]	x1	x2
1.0	0.0	19.945536387662504	3.85214120038979
1.0	0.0	20.674308066353493	4.098585619118175
0.0	1.0	20.346647025958433	2.7107604387194626
0.0	1.0	18.699653829045985	5.2111542692543065
1.0	0.0	21.51851187887476	2.432390426907621

CHAPTER

TWO

BIBLIOGRAPHY

CHAPTER

THREE

PYSPARK FORMULA

3.1 Formula

The `formula` module contains code to extract values from a record (e.g. a Spark dataframe Record) based on the model definition.

class ydot.formula.**CatExtractor**(*record*, *term*)
Bases: *ydot.formula.Extractor*

Categorical extractor (no levels).

__init__(*record*, *term*)
ctor.

Parameters

- **record** – Dictionary.
- **term** – Model term.

Returns None.

property value
Gets the extracted value.

class ydot.formula.**ConExtractor**(*record*, *term*)
Bases: *ydot.formula.Extractor*

Continuous extractor (no functions).

__init__(*record*, *term*)
ctor.

Parameters

- **record** – Dictionary.
- **term** – Model term.

Returns None.

property value
Gets the extracted value.

class ydot.formula.**Extractor**(*record*, *term*, *term_type*)
Bases: abc.ABC

Extractor to get value based on model term.

__init__(*record*, *term*, *term_type*)
ctor.

Param Dictionary.

Term Model term.

Term_type Type of term.

Returns None

abstract property value
Gets the extracted value.

class ydot.formula.**FunExtractor**(*record*, *term*)
Bases: *ydot.formula.Extractor*

Continuous extractor (with functions defined).

__init__(*record*, *term*)
ctor.

Parameters

- **record** – Dictionary.
- **term** – Model term.

Returns None.

property value
Gets the extracted value.

class ydot.formula.**IntExtractor**(*record*, *term*)
Bases: *ydot.formula.Extractor*

Intercept extractor. Always returns 1.0.

__init__(*record*, *term*)
ctor.

Parameters

- **record** – Dictionary.
- **term** – Model term.

Returns None.

property value
Gets the extracted value.

class ydot.formula.**InteractionExtractor**(*record*, *terms*)
Bases: object

Interaction extractor for interaction effects.

__init__(*record*, *terms*)
ctor.

Parameters

- **record** – Dictionary.
- **terms** – Model term (possibly with interaction effects).

Returns None.

property value

class ydot.formula.**LvlExtractor**(*record*, *term*)
Bases: *ydot.formula.Extractor*

Categorical extractor (with levels).

__init__(*record*, *term*)
ctor.

Parameters

- **record** – Dictionary.
- **term** – Model term.

Returns None.

property value
Gets the extracted value.

class ydot.formula.**TermEnum**(*value*)
Bases: enum.IntEnum

Term types.

- CAT: categorical without levels specified
- LVL: categorical with levels specified
- CON: continuous
- FUN: continuous with function transformations
- INT: intercept

CAT = 1

CON = 3

FUN = 4

INT = 5

LVL = 2

static get_extractor(*record*, *term*)
Gets the associated extractor based on the specified term.

Parameters

- **record** – Dictionary.
- **term** – Model term.

Returns Extractor.

3.2 Spark

The spark module contains code to transform a Spark dataframe into design matrices as specified by a formula.

ydot.spark.**get_columns**(*formula*, *sdf*, *profile=None*)
Gets the expanded columns of the specified Spark dataframe using the specified formula.

Parameters

- **formula** – Formula (R-like, based on patsy).

- **sdf** – Spark dataframe.
- **profile** – Profile. Default is *None* and profile will be determined empirically.

Returns Tuple of columns for y, X.

ydot.spark.**get_profile**(*sdf*)
Gets the field profiles of the specified Spark dataframe.

Parameters **sdf** – Spark dataframe.

Returns Dictionary.

ydot.spark.**smatrices**(*formula*, *sdf*, *profile=None*)
Gets tuple of design/model matrices.

Parameters

- **formula** – Formula.
- **sdf** – Spark dataframe.
- **profile** – Dictionary of data profile.

Returns y, X Spark dataframes.

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

CHAPTER

FIVE

ABOUT

One-Off Coder is an educational, service and product company. Please visit us online to discover how we may help you achieve life-long success in your personal coding career or with your company's business goals and objectives.

-
-
-
-
-
-

CHAPTER

SIX

COPYRIGHT

6.1 Documentation

6.2 Software

```
Copyright 2020 One-Off Coder

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

   http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

6.3 Art

```
Copyright 2020 Daytchia Vang
```

CHAPTER

SEVEN

CITATION

```
@misc{oneoffcoder_ydot_2020,
title={ydot, R-like formulas for Spark Dataframes},
url={https://github.com/oneoffcoder/pyspark-formula},
author={Jee Vang},
year={2020},
month={Dec}}
```

CHAPTER

EIGHT

AUTHOR

Jee Vang, Ph.D.

-

CHAPTER

NINE

HELP

-
-

BIBLIOGRAPHY

[pat] patsy. Patsy - describing statistical models in python. URL: https://patsy.readthedocs.io/en/latest/index.html.

PYTHON MODULE INDEX

y

INDEX

Symbols

C

E

F

G

I

L

M

S

T

V

Y