
PySpark Data Audit



PySparkAudit: PySpark Data Audit

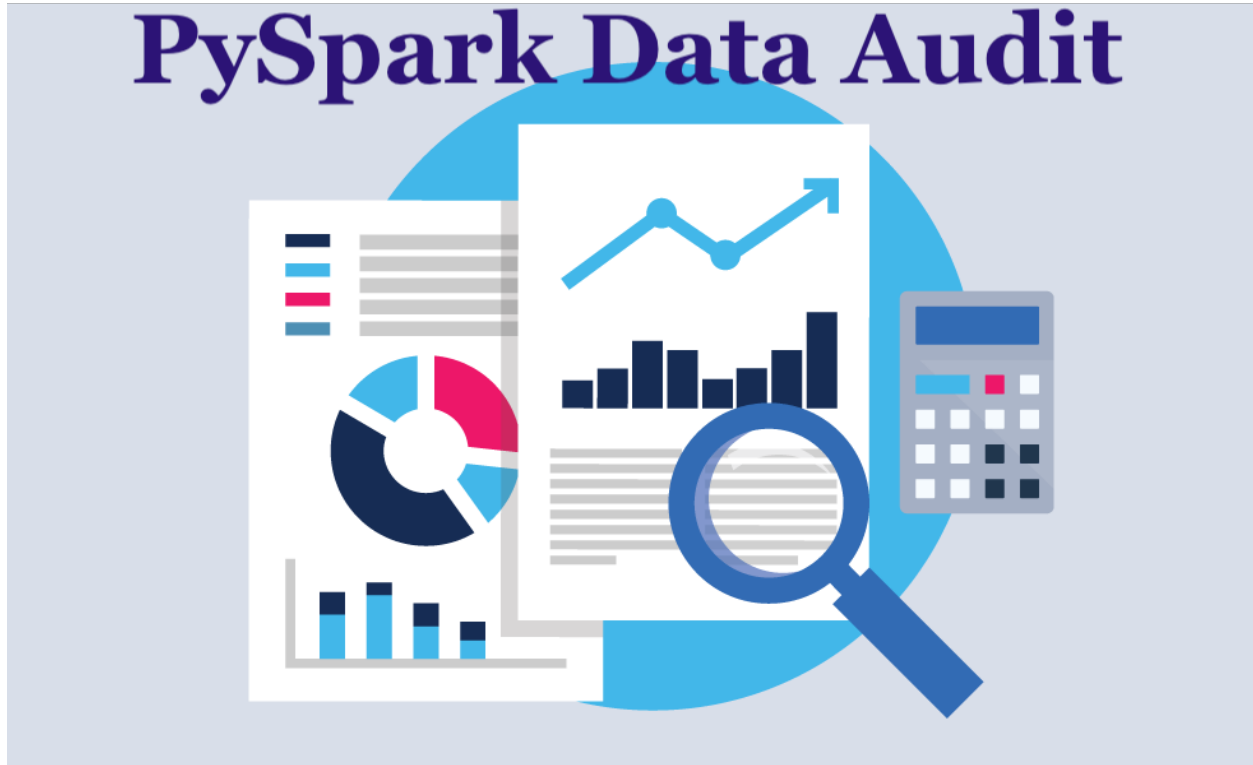
Wenqiang Feng and Yiming Xu

July 03, 2019

CONTENTS

1	Preface	3
1.1	About	3
1.1.1	About this API	3
1.1.2	About the author	3
1.2	Acknowledgement	4
1.3	Feedback and suggestions	4
2	How to Install	5
2.1	Install with <code>pip</code>	5
2.2	Install from Repo	5
2.2.1	Clone the Repository	5
2.2.2	Install	5
2.3	Uninstall	5
2.4	Test	6
2.4.1	Run test code	6
2.4.2	Audited Results	6
3	PySpark Data Audit Functions	9
3.1	Basic Functions	9
3.1.1	<code>mkdir</code>	9
3.1.2	<code>mkdir_clean</code>	9
3.1.3	<code>df_merge</code>	9
3.1.4	<code>data_types</code>	10
3.1.5	<code>dtypes_class</code>	10
3.1.6	<code>counts</code>	11
3.1.7	<code>describe</code>	12
3.1.8	<code>percentiles</code>	13
3.1.9	<code>feature_len</code>	14
3.1.10	<code>freq_items</code>	15
3.1.11	<code>rates</code>	16
3.1.12	<code>corr_matrix</code>	17
3.2	Plot Functions	18
3.2.1	<code>hist_plot</code>	18

3.2.2	bar_plot	18
3.2.3	trend_plot	19
3.3	Summary Functions	19
3.3.1	dataset_summary	19
3.3.2	numeric_summary	20
3.3.3	category_summary	20
3.4	Auditing Function	21
3.4.1	auditing	21
3.5	Plotting Function	22
3.5.1	fig_plots	22
4	Auditing Demos	23
4.1	Auditing function by function	23
4.2	Auditing in one function	25
4.2.1	print in bash	27
4.2.2	Audited results folder	28
4.3	Auditing on Big Dataset	33
4.3.1	print in bash	35
4.3.2	Audited results folder	36
5	Main Reference	39
	Bibliography	41



Welcome to our **PySparkAudit: PySpark Data Audit Library API**! The PDF version can be downloaded from [HERE](#).

You can install the PySparkAudit from [PyPI](<https://pypi.org/project/PySparkAudit>):

```
pip install PySparkAudit
```


PREFACE

Chinese proverb

Good tools are prerequisite to the successful execution of a job. – old Chinese proverb

1.1 About

1.1.1 About this API

This document is the API book for our **PySparkAudit**: PySpark Data Audit Library [[PySparkAudit](#)] API. The PDF version can be downloaded from [HERE](#). **You may download and distribute it. Please be aware, however, that the note contains typos as well as inaccurate or incorrect description.**

The API assumes that the reader has a preliminary knowledge of python programing and Linux. And this document is generated automatically by using [sphinx](#).

The python version **PyAudit**: Python Data Audit Library API can be found at [[PyAudit](#)].

1.1.2 About the author

- **Wenqiang Feng**
 - Data Scientist and PhD in Mathematics
 - University of Tennessee at Knoxville
 - Webpage: <http://web.utk.edu/~wfeng1/>
 - Email: von198@gmail.com
- **Yiming Xu**

- Data Scientist and Master of Data Science
- Harvard University
- Email: yimingxu@g.harvard.edu

- **Biography**

Wenqiang Feng is Data Scientist within DST's Applied Analytics Group. Dr. Feng's responsibilities include providing DST clients with access to cutting-edge skills and technologies, including Big Data analytic solutions, advanced analytic and data enhancement techniques and modeling.

Dr. Feng has deep analytic expertise in data mining, analytic systems, machine learning algorithms, business intelligence, and applying Big Data tools to strategically solve industry problems in a cross-functional business. Before joining DST, Dr. Feng was an IMA Data Science Fellow at The Institute for Mathematics and its Applications (IMA) at the University of Minnesota. While there, he helped startup companies make marketing decisions based on deep predictive analytics.

Dr. Feng graduated from University of Tennessee, Knoxville, with Ph.D. in Computational Mathematics and Master's degree in Statistics. He also holds Master's degree in Computational Mathematics from Missouri University of Science and Technology (MST) and Master's degree in Applied Mathematics from the University of Science and Technology of China (USTC).

- **Declaration**

The work of Wenqiang Feng was supported by the IMA, while working at IMA. However, any opinion, finding, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the IMA, UTK, DST and Harvard University.

1.2 Acknowledgement

At here, Wenqiang Feng would like to thank **Weiyu Wang** at Missouri University of Science and Technology and **Jiangtao (Lotto) Xie** at Purdue University for the unit testing and valuable discussion.

1.3 Feedback and suggestions

Your comments and suggestions are highly appreciated. I am more than happy to receive corrections, suggestions or feedbacks through email (Wenqiang Feng: von198@gmail.com and Yiming Xu: yimingxu@g.harvard.edu) for improvements.

HOW TO INSTALL

2.1 Install with `pip`

You can install the `PySparkAudit` from [PyPI](<https://pypi.org/project/PySparkAudit>):

```
pip install PySparkAudit
```

2.2 Install from Repo

2.2.1 Clone the Repository

```
git clone https://github.com/runawayhorse001/PySparkAudit.git
```

2.2.2 Install

```
cd PySparkAudit
pip install -r requirements.txt
python setup.py install
```

2.3 Uninstall

```
pip uninstall statspy
```

2.4 Test

2.4.1 Run test code

```
cd PySparkAudit/test
python test.py
```

test.py

```
from pyspark.sql import SparkSession

spark = SparkSession \
    .builder \
    .appName("Python Spark regression example") \
    .config("spark.some.config.option", "some-value") \
    .getOrCreate()

# from PySparkAudit import dtypes_class, hist_plot, bar_plot, freq_
→items, feature_len
# from PySparkAudit import dataset_summary, rates, trend_plot

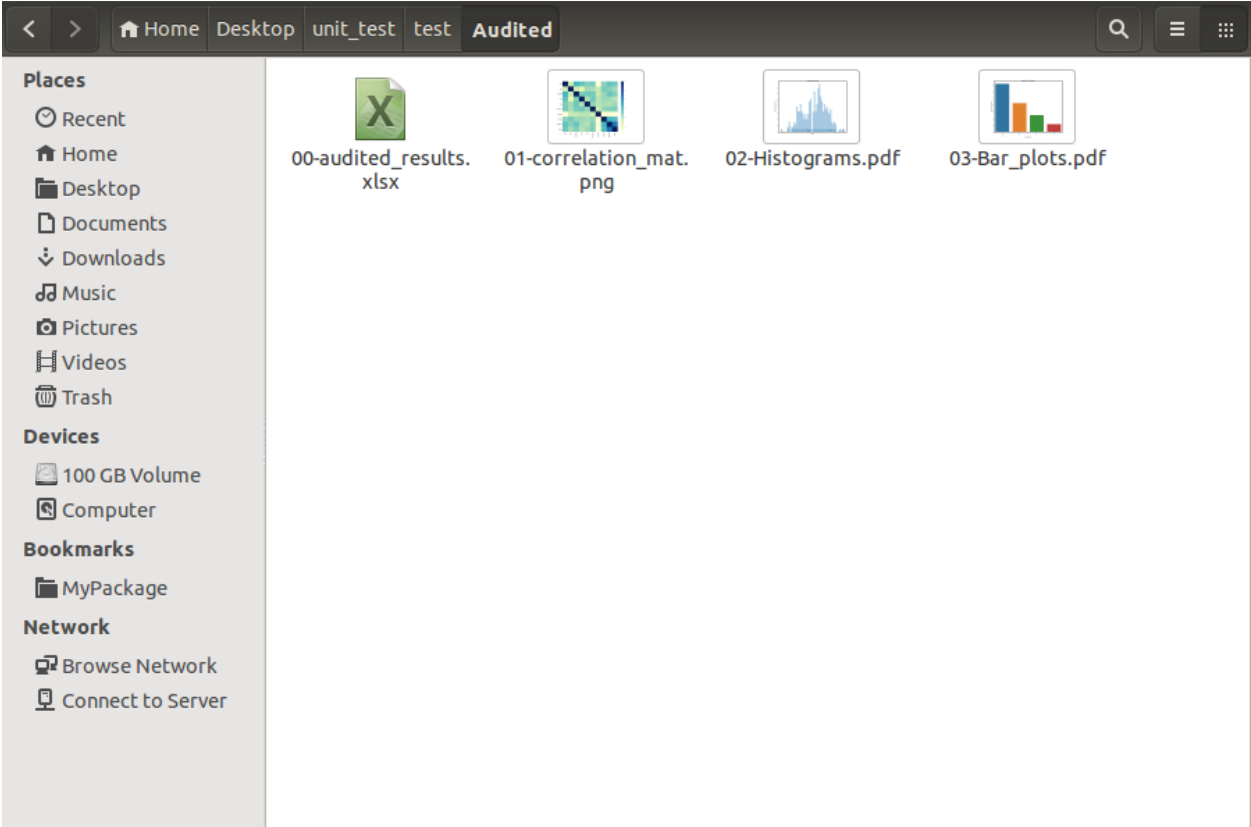
# path = '/home/feng/Desktop'

# import PySpark Audit function
from PySparkAudit import auditing

# load dataset
data = spark.read.csv(path='Heart.csv',
                      sep=',', encoding='UTF-8', comment=None,
→header=True, inferSchema=True)

# auditing in one function
print(auditing(data, display=True))
```

2.4.2 Audited Results



PYSPARK DATA AUDIT FUNCTIONS

3.1 Basic Functions

3.1.1 mkdir

`PySparkAudit.PySparkAudit.mkdir(path)`

Make a new directory. if it's exist, keep the old files.

Parameters `path` – the directory path

3.1.2 mkdir_clean

`PySparkAudit.PySparkAudit.mkdir_clean(path)`

Make a new directory. if it's exist, remove the old files.

Parameters `path` – the directory path

3.1.3 df_merge

`PySparkAudit.PySparkAudit.df_merge(dfs, key, how='left')`

Merge multiple pandas data frames with same key.

Parameters

- **dfs** – name list of the data frames
- **key** – key for join
- **how** – method for join, the default value is left

Returns merged data frame

3.1.4 data_types

`PySparkAudit.PySparkAudit.data_types(df_in, tracking=False)`

Generate the data types of the rdd data frame.

Parameters

- **df_in** – the input rdd data frame
- **tracking** – the flag for displaying CPU time, the default value is False

Returns data types pandas data frame

```
>>> test = spark.createDataFrame([
    ('Joe', 67, 'F', 7000, 'asymptomatic', 286.1,
    ↪ '2019-6-28'),
    ('Henry', 67, 'M', 8000, 'asymptomatic', 229.2,
    ↪ '2019-6-29'),
    ('Sam', 37, 'F', 6000, 'nonanginal', 250.3,
    ↪ '2019-6-30'),
    ('Max', 56, 'M', 9000, 'nontypical', 236.4,
    ↪ '2019-5-28'),
    ('Mat', 56, 'F', 9000, 'asymptomatic', 254.5,
    ↪ '2019-4-28')],
    ['Name', 'Age', 'Sex', 'Salary', 'ChestPain',
    ↪ 'Chol', 'CreatDate'])
>>> test = test.withColumn('CreatDate', F.col('CreatDate').cast(
    ↪ 'timestamp'))
>>> from PySparkAudit import data_types
>>> data_types(test)
   feature      dtypes
0      Name    string
1      Age    bigint
2      Sex    string
3    Salary    bigint
4 ChestPain    string
5      Chol    double
6 CreatDate timestamp
```

3.1.5 dtypes_class

`PySparkAudit.PySparkAudit.dtypes_class(df_in)`

Generate the data type categories: numerical, categorical, date and unsupported category.

Parameters **df_in** – the input rdd data frame

Returns data type categories

```

>>> test = spark.createDataFrame([
    ('Joe', 67, 'F', 7000, 'asymptomatic', 286.1,
    ↪ '2019-6-28'),
    ('Henry', 67, 'M', 8000, 'asymptomatic', 229.2,
    ↪ '2019-6-29'),
    ('Sam', 37, 'F', 6000, 'nonanginal', 250.3,
    ↪ '2019-6-30'),
    ('Max', 56, 'M', 9000, 'nontypical', 236.4,
    ↪ '2019-5-28'),
    ('Mat', 56, 'F', 9000, 'asymptomatic', 254.5,
    ↪ '2019-4-28')],
    ['Name', 'Age', 'Sex', 'Salary', 'ChestPain',
    ↪ 'Chol', 'CreatDate'])
>>> test = test.withColumn('CreatDate', F.col('CreatDate').cast(
    ↪ 'timestamp'))
>>> from PySparkAudit import dtypes_class
>>> dtypes_class(test)
(      feature      DataType
0      Name      StringType
1      Age      LongType
2      Sex      StringType
3      Salary      LongType
4      ChestPain      StringType
5      Chol      DoubleType
6      CreatDate      TimestampType,
['Age', 'Salary', 'Chol'],
['Name', 'Sex', 'ChestPain'],
['CreatDate'], [])

```

3.1.6 counts

`PySparkAudit.PySparkAudit.counts(df_in, tracking=False)`

Generate the row counts and not null rows and distinct counts for each feature.

Parameters

- **df_in** – the input rdd data frame
- **tracking** – the flag for displaying CPU time, the default value is False

Returns the counts in pandas data frame

```

>>> test = spark.createDataFrame([
    ('Joe', None, 'F', 70000, 'asymptomatic', 286.
    ↪1, '2019-6-28'),

```

(continues on next page)

(continued from previous page)

```

                ('Henry', 67, 'M', 80000, 'asymptomatic', 229.
→2, '2019-6-29'),
                ('Sam', 37, 'F', 60000, 'nonanginal', 250.3,
→'2019-6-30'),
                ('Max', 56, ' ', 90000, None, 236.4, '2019-5-
→28'),
                ('Mat', 56, 'F', None, 'asymptomatic', 254.5,
→'2019-4-28')],
                ['Name', 'Age', 'Sex', 'Salary', 'ChestPain',
→'Chol', 'CreatDate']
            )
>>> test = test.withColumn('CreatDate', F.col('CreatDate').cast(
→'timestamp'))
>>> from PySparkAudit import counts
>>> counts(test)

```

	feature	row_count	notnull_count	distinct_count
0	Name	5	5	5
1	Age	5	4	3
2	Sex	5	5	3
3	Salary	5	4	4
4	ChestPain	5	4	2
5	Chol	5	5	5
6	CreatDate	5	5	5

3.1.7 describe

PySparkAudit.PySparkAudit.**describe**(*df_in*, *columns=None*, *tracking=False*)

Generate the simple data frame description using *.describe()* function in pyspark.

Parameters

- **df_in** – the input rdd data frame
- **columns** – the specific feature columns, the default value is None
- **tracking** – the flag for displaying CPU time, the default value is False

Returns the description in pandas data frame

```

>>> test = spark.createDataFrame([
                ('Joe', 67, 'F', 7000, 'asymptomatic', 286.1,
→'2019-6-28'),
                ('Henry', 67, 'M', 8000, 'asymptomatic', 229.2,
→'2019-6-29'),
                ('Sam', 37, 'F', 6000, 'nonanginal', 250.3, '2019-
→6-30'),

```

(continues on next page)

(continued from previous page)

```

        ('Max', 56, 'M', 9000, 'nontypical', 236.4, '2019-
→5-28'),
        ('Mat', 56, 'F', 9000, 'asymptomatic', 254.5,
→'2019-4-28')],
        ['Name', 'Age', 'Sex', 'Salary', 'ChestPain', 'Chol
→', 'CreatDate']
    )
>>> test = test.withColumn('CreatDate', F.col('CreatDate').cast(
→'timestamp'))
>>> from PySparkAudit import describe
>>> describe(test)
summary    count      mean      ...      min      max
feature
Name         5      None      ...      Henry      Sam
Age          5      56.6      ...         37         67
Sex          5      None      ...         F         M
Salary       5  78000.0      ...      60000      90000
ChestPain    5      None      ...  asymptomatic  nontypical
Chol         5      251.3      ...      229.2      286.1
CreatDate    5      None      ...  2019-4-28  2019-6-30

```

[7 rows x 5 columns]

3.1.8 percentiles

PySparkAudit.PySparkAudit.**percentiles** (*df_in*, *deciles=False*, *tracking=False*)

Generate the percentiles for rdd data frame.

Parameters

- **df_in** – the input rdd data frame
- **deciles** – the flag for generate the deciles
- **tracking** – the flag for displaying CPU time, the default value is False

Returns percentiles in pandas data frame

```

>>> test = spark.createDataFrame([
        ('Joe', 67, 'F', 7000, 'asymptomatic', 286.1,
→'2019-6-28'),
        ('Henry', 67, 'M', 8000, 'asymptomatic', 229.2,
→ '2019-6-29'),
        ('Sam', 37, 'F', 6000, 'nonanginal', 250.3,
→'2019-6-30'),

```

(continues on next page)

(continued from previous page)

```

        ('Max', 56, 'M', 9000, 'nontypical', 236.4,
→ '2019-5-28'),
        ('Mat', 56, 'F', 9000, 'asymptomatic', 254.5,
→ '2019-4-28')],
        ['Name', 'Age', 'Sex', 'Salary', 'ChestPain',
→ 'Chol', 'CreatDate']
    )
>>> from PySparkAudit import percentiles
>>> percentiles(test)
  feature      Q1      Med      Q3
0     Age    56.0    67.0    67.0
1  Salary 80000.0 90000.0 90000.0
2     Chol   250.3   254.5   286.1

```

3.1.9 feature_len

PySparkAudit.PySparkAudit.**feature_len** (*df_in*, *tracking=False*)

Generate feature length statistical results for each feature in the rdd data frame.

Parameters

- **df_in** – the input rdd data frame
- **tracking** – the flag for displaying CPU time, the default value is False

Returns the feature length statistical results in pandas data frame

```

>>> test = spark.createDataFrame([
        ('Joe', 67, 'F', 7000, 'asymptomatic', 286.1,
→ '2019-6-28'),
        ('Henry', 67, 'M', 8000, 'asymptomatic', 229.2,
→ '2019-6-29'),
        ('Sam', 37, 'F', 6000, 'nonanginal', 250.3,
→ '2019-6-30'),
        ('Max', 56, 'M', 9000, 'nontypical', 236.4,
→ '2019-5-28'),
        ('Mat', 56, 'F', 9000, 'asymptomatic', 254.5,
→ '2019-4-28')],
        ['Name', 'Age', 'Sex', 'Salary', 'ChestPain',
→ 'Chol', 'CreatDate']
    )
>>> from PySparkAudit import feature_len
>>> feature_len(test)
  feature  min_length  avg_length  max_length
0     Name          3.0          3.4          5.0

```

(continues on next page)

(continued from previous page)

1	Age	2.0	2.0	2.0
2	Sex	1.0	1.0	1.0
3	Salary	5.0	5.0	5.0
4	ChestPain	10.0	11.2	12.0
5	Chol	5.0	5.0	5.0
6	CreatDate	9.0	9.0	9.0

3.1.10 freq_items

PySparkAudit.PySparkAudit.**freq_items** (*df_in*, *top_n=5*, *tracking=False*)

Generate the top_n frequent items in for each feature in the rdd data frame.

Parameters

- **df_in** – the input rdd data frame
- **top_n** – the number of the most frequent item
- **tracking** – the flag for displaying CPU time, the default value is False

Returns

```
>>> test = spark.createDataFrame([
    ('Joe', 67, 'F', 7000, 'asymptomatic', 286.1,
    ↪ '2019-6-28'),
    ('Henry', 67, 'M', 8000, 'asymptomatic', 229.2,
    ↪ '2019-6-29'),
    ('Sam', 37, 'F', 6000, 'nonanginal', 250.3,
    ↪ '2019-6-30'),
    ('Max', 56, 'M', 9000, 'nontypical', 236.4,
    ↪ '2019-5-28'),
    ('Mat', 56, 'F', 9000, 'asymptomatic', 254.5,
    ↪ '2019-4-28')],
    ['Name', 'Age', 'Sex', 'Salary', 'ChestPain',
    ↪ 'Chol', 'CreatDate'])
>>> from PySparkAudit import freq_items
>>> freq_items(test)
feature                                freq_items[value, freq]
0      Name  [[Joe, 1], [Mat, 1], [Henry, 1], [Sam, 1], [Ma...
1      Age    [[67, 2], [56, 2], [37, 1]]
2      Sex    [[F, 3], [M, 2]]
3      Salary  [[90000, 2], [70000, 1], [80000, 1], [60000, 1]]
4      ChestPain  [[asymptomatic, 3], [nontypical, 1], [nonangin...
5      Chol    [[286.1, 1], [250.3, 1], [229.2, 1], [236.4, 1...
6      CreatDate  [[2019-6-30, 1], [2019-5-28, 1], [2019-4-28, 1...]
```

3.1.11 rates

`PySparkAudit.PySparkAudit.rates(df_in, columns=None, numeric=True, tracking=False)`

Generate the null, empty, negative, zero and positive value rates and feature variance for each feature in the rdd data frame.

Parameters

- **df_in** – the input rdd data frame
- **columns** – the specific feature columns, the default value is None
- **numeric** – the flag for numerical rdd data frame, the default value is True
- **tracking** – the flag for displaying CPU time, the default value is False

Returns the null, empty, negative, zero and positive value rates and feature variance in pandas data frame

```
>>> test = spark.createDataFrame([
    ('Joe', 67, 'F', 7000, 'asymptomatic', 286.1,
→ '2019-6-28'),
    ('Henry', 67, 'M', 8000, 'asymptomatic', 229.2,
→ '2019-6-29'),
    ('Sam', 37, 'F', 6000, 'nonanginal', 250.3,
→ '2019-6-30'),
    ('Max', 56, 'M', 9000, 'nontypical', 236.4,
→ '2019-5-28'),
    ('Mat', 56, 'F', 9000, 'asymptomatic', 254.5,
→ '2019-4-28')],
    ['Name', 'Age', 'Sex', 'Salary', 'ChestPain',
→ 'Chol', 'CreatDate'])
>>> from PySparkAudit import rates
>>> rates(test)
```

	feature	feature_variance	...	rate_zero	rate_pos
0	Age	0.6	...	0.0	1.0
1	Salary	0.8	...	0.0	1.0
2	Chol	1.0	...	0.0	1.0
3	Name	1.0	...	0.0	0.0
4	Sex	0.4	...	0.0	0.0
5	ChestPain	0.6	...	0.0	0.0
6	CreatDate	1.0	...	0.0	0.0

[7 rows x 7 columns]

3.1.12 corr_matrix

`PySparkAudit.PySparkAudit.corr_matrix(df_in, method='pearson', output_dir=None, rotation=True, display=False, tracking=False)`

Generate the correlation matrix and heat map plot for rdd data frame.

Parameters

- **df_in** – the input rdd data frame
- **method** – the method which applied to calculate the correlation matrix: pearson or spearman. the default value is pearson
- **output_dir** – the out put directory, the default value is the current working directory
- **rotation** – the flag for rotating the xticks in the plot, the default value is True
- **display** – the flag for displaying the figures, the default value is False
- **tracking** – the flag for displaying CPU time, the default value is False

Returns the correlation matrix in pandas data frame

```
>>> test = spark.createDataFrame([
    ('Joe', 67, 'F', 7000, 'asymptomatic', 286.1,
→ '2019-6-28'),
    ('Henry', 67, 'M', 8000, 'asymptomatic', 229.2,
→ '2019-6-29'),
    ('Sam', 37, 'F', 6000, 'nonanginal', 250.3,
→ '2019-6-30'),
    ('Max', 56, 'M', 9000, 'nontypical', 236.4,
→ '2019-5-28'),
    ('Mat', 56, 'F', 9000, 'asymptomatic', 254.5,
→ '2019-4-28')],
    ['Name', 'Age', 'Sex', 'Salary', 'ChestPain',
→ 'Chol', 'CreatDate'])
>>> from PySparkAudit import corr_matrix
>>> corr_matrix(test)
=====
The correlation matrix plot Corr.png was located at:
/home/feng/Audited
      Age      Salary      Chol
Age      1.000000  0.431663  0.147226
Salary    0.431663  1.000000 -0.388171
Chol      0.147226 -0.388171  1.000000
```

3.2 Plot Functions

3.2.1 hist_plot

```
PySparkAudit.PySparkAudit.hist_plot(df_in, bins=50, output_dir=None,  
                                       sample_size=None, display=False,  
                                       tracking=False)
```

Histogram plot for the numerical features in the rdd data frame. **This part is super time and memory consuming.** If the data size is larger than 10,000, the histograms will be saved in .pdf format. Otherwise, the histograms will be saved in .png format in hist folder.

If your time and memory are limited, you can use sample_size to generate the subset of the data frame to generate the histograms.

Parameters

- **df_in** – the input rdd data frame
- **bins** – the number of bins for generate the bar plots
- **output_dir** – the out put directory, the default value is the current working directory
- **sample_size** – the size for generate the subset from the rdd data frame, the default value none
- **display** – the flag for displaying the figures, the default value is False
- **tracking** – the flag for displaying CPU time, the default value is False

3.2.2 bar_plot

```
PySparkAudit.PySparkAudit.bar_plot(df_in, top_n=20, rotation=True, out-  
                                       put_dir=None, display=False, track-  
                                       ing=False)
```

Bar plot for the categorical features in the rdd data frame.

Parameters

- **df_in** – the input rdd data frame
- **top_n** – the number of the most frequent feature to show in the bar plot
- **rotation** – the flag for rotating the xticks in the plot, the default value is True
- **output_dir** – the out put directory, the default value is the current working directory
- **display** – the flag for displaying the figures, the default value is False

- **tracking** – the flag for displaying CPU time, the default value is False

3.2.3 trend_plot

`PySparkAudit.PySparkAudit.trend_plot(df_in, types='day', d_time=None, rotation=True, output_dir=None, display=False, tracking=False)`

Trend plot for the aggregated time series data if the rdd data frame has date features and numerical features.

Parameters

- **df_in** – the input rdd data frame
- **types** – the types for time feature aggregation: day, month, year, the default value is day
- **d_time** – the specific feature name of the date feature, the default value is the first date feature in the rdd data frame
- **rotation** – the flag for rotating the xticks in the plot, the default value is True
- **output_dir** – the out put directory, the default value is the current working directory
- **display** – the flag for displaying the figures, the default value is False
- **tracking** – the flag for displaying CPU time, the default value is False

3.3 Summary Functions

3.3.1 dataset_summary

`PySparkAudit.PySparkAudit.dataset_summary(df_in, tracking=False)`

The data set basics summary.

Parameters

- **df_in** – the input rdd data frame
- **tracking** – the flag for displaying CPU time, the default value is False

Returns data set summary in pandas data frame

3.3.2 numeric_summary

```
PySparkAudit.PySparkAudit.numeric_summary(df_in, columns=None,  
                                           deciles=False, top_n=5,  
                                           tracking=False)
```

The auditing function for numerical rdd data frame.

Parameters

- **df_in** – the input rdd data frame
- **columns** – the specific feature columns, the default value is None
- **deciles** – the flag for generate the deciles
- **top_n** – the number of the most frequent item
- **tracking** – the flag for displaying CPU time, the default value is False

Returns the audited results for the numerical features in pandas data frame

3.3.3 category_summary

```
PySparkAudit.PySparkAudit.category_summary(df_in, columns=None,  
                                           top_n=5, tracking=False)
```

The auditing function for categorical rdd data frame.

Parameters

- **df_in** – the input rdd data frame
- **columns** – the specific feature columns, the default value is None
- **top_n** – the number of the most frequent item
- **tracking** – the flag for displaying CPU time, the default value is False

Returns the audited results for the categorical features in pandas data frame

3.4 Auditing Function

3.4.1 auditing

```
PySparkAudit.PySparkAudit.auditing(df_in, writer=None, columns=None,  
                                     deciles=False, top_freq_item=5,  
                                     bins=50, top_cat_item=20,  
                                     method='pearson', output_dir=None,  
                                     types='day', d_time=None, ro-  
                                     tation=True, sample_size=None,  
                                     display=False, tracking=False)
```

The wrapper of auditing functions.

Parameters

- **df_in** – the input rdd data frame
- **writer** – the writer for excel output
- **columns** – the specific feature columns, the default value is None
- **deciles** – the flag for generate the deciles
- **top_freq_item** – the number of the most frequent item
- **bins** – the number of bins for generate the bar plots
- **top_cat_item** – the number of the most frequent feature to show in the bar plot
- **method** – the method which applied to calculate the correlation matrix: pearson or spearman. the default value is pearson
- **output_dir** – the out put directory, the default value is the current working directory
- **types** – the types for time feature aggregation: day, month, year, the default value is day
- **d_time** – the specific feature name of the date feature, the default value is the first date feature in the rdd data frame
- **rotation** – the flag for rotating the xticks in the plot, the default value is True
- **sample_size** – the size for generate the subset from the rdd data frame, the default value none
- **display** – the flag for displaying the figures, the default value is False
- **tracking** – the flag for displaying CPU time, the default value is False

Returns the all audited results in pandas data frame

3.5 Plotting Function

3.5.1 fig_plots

```
PySparkAudit.PySparkAudit.fig_plots(df_in, output_dir=None,  
                                       bins=50, top_n=20, types='day',  
                                       d_time=None, rotation=True, sam-  
                                       ple_size=None, display=False,  
                                       tracking=False)
```

The wrapper for the plot functions.

Parameters

- **df_in** – the input rdd data frame
- **output_dir** – the out put directory, the default value is the current working directory
- **bins** – the number of bins for generate the bar plots
- **top_n** – the number of the most frequent feature to show in the bar plot
- **types** – the types for time feature aggregation: day, month, year, the default value is day
- **d_time** – the specific feature name of the date feature, the default value is the first date feature in the rdd data frame
- **rotation** – the flag for rotating the xticks in the plot, the default value is True
- **sample_size** – the size for generate the subset from the rdd data frame, the default value none
- **display** – the flag for displaying the figures, the default value is False
- **tracking** – the flag for displaying CPU time, the default value is False

AUDITING DEMOS

The following demos are designed to show how to use `PySparkAudit` to audit `rdd DataFrame`.

4.1 Auditing function by function

If you just need a piece of the audit result, you can call the corresponding function to generate it. There are 9 basic auditing functions, 3 figure plot functions and 3 summary functions in the `PySparkAudit` library.

syntax

```
from PySparkAudit import *
```

1. Basic Functions:

- a. `data_types: PySparkAudit.data_types`
- b. `dtypes_class: PySparkAudit.dtypes_class`
- c. `dtypes_class: PySparkAudit.counts`
- d. `dtypes_class: PySparkAudit.describe`
- e. `dtypes_class: PySparkAudit.percentiles`
- f. `dtypes_class: PySparkAudit.feature_len`
- g. `dtypes_class: PySparkAudit.freq_items`
- h. `dtypes_class: PySparkAudit.rates`
- i. `dtypes_class: PySparkAudit.corr_matrix`

2. Plot Functions:

- a. `hist_plot: PySparkAudit.hist_plot`
- b. `bar_plot: PySparkAudit.bar_plot`

c. `trend_plot`: `PySparkAudit.trend_plot`

3. Summary Functions

a. `dataset_summary`: `PySparkAudit.dataset_summary`

b. `numeric_summary`: `PySparkAudit.numeric_summary`

c. `category_summary`: `PySparkAudit.category_summary`

For example:

```
from pyspark.sql import SparkSession

spark = SparkSession \
    .builder \
    .appName("Python Spark regression example") \
    .config("spark.some.config.option", "some-value") \
    .getOrCreate()

# import PySpark Audit functions
from PySparkAudit import data_types, hist_plot, bar_plot, freq_items,
    feature_len
from PySparkAudit import dataset_summary, rates
from PySparkAudit import trend_plot, auditing

# load dataset
data = spark.read.csv(path='Heart.csv',
                      sep=',', encoding='UTF-8', comment=None,
    header=True, inferSchema=True)

# audit function by function

# data types
print(data_types(data))

# check frequent items
print(freq_items(data))

# bar plot for categorical features
bar_plot(data, display=True)
```

Result:

	feature	dtypes
0	Age	int
1	Sex	int
2	ChestPain	string

(continues on next page)

(continued from previous page)

```

3      RestBP      int
4      Chol       int
5      Fbs        int
6      RestECG    int
7      MaxHR      int
8      ExAng      int
9      Oldpeak    double
10     Slope      int
11     Ca         string
12     Thal       string
13     AHD        string

feature                                     freq_items[value, freq]
0      Age      [[58, 19], [57, 17], [54, 16], [59, 14], [52, ...
1      Sex      [[1, 206], [0, 97]]
2      ChestPain [[asymptomatic, 144], [nonanginal, 86], [nonty...
3      RestBP   [[120, 37], [130, 36], [140, 32], [110, 19], [...
4      Chol     [[197, 6], [234, 6], [204, 6], [254, 5], [212,...
5      Fbs      [[0, 258], [1, 45]]
6      RestECG  [[0, 151], [2, 148], [1, 4]]
7      MaxHR    [[162, 11], [163, 9], [160, 9], [152, 8], [132...
8      ExAng    [[0, 204], [1, 99]]
9      Oldpeak  [[0.0, 99], [1.2, 17], [0.6, 14], [1.0, 14], [...
10     Slope    [[1, 142], [2, 140], [3, 21]]
11     Ca       [[0, 176], [1, 65], [2, 38], [3, 20], [NA, 4]]
12     Thal     [[normal, 166], [reversible, 117], [fixed, 18]...
13     AHD      [[No, 164], [Yes, 139]]
=====
The Bar plot Bar_plots.pdf was located at:
/home/feng/Dropbox/MyTutorial/PySparkAudit/test/Audited

Process finished with exit code 0

```

and

4.2 Auditing in one function

For example:

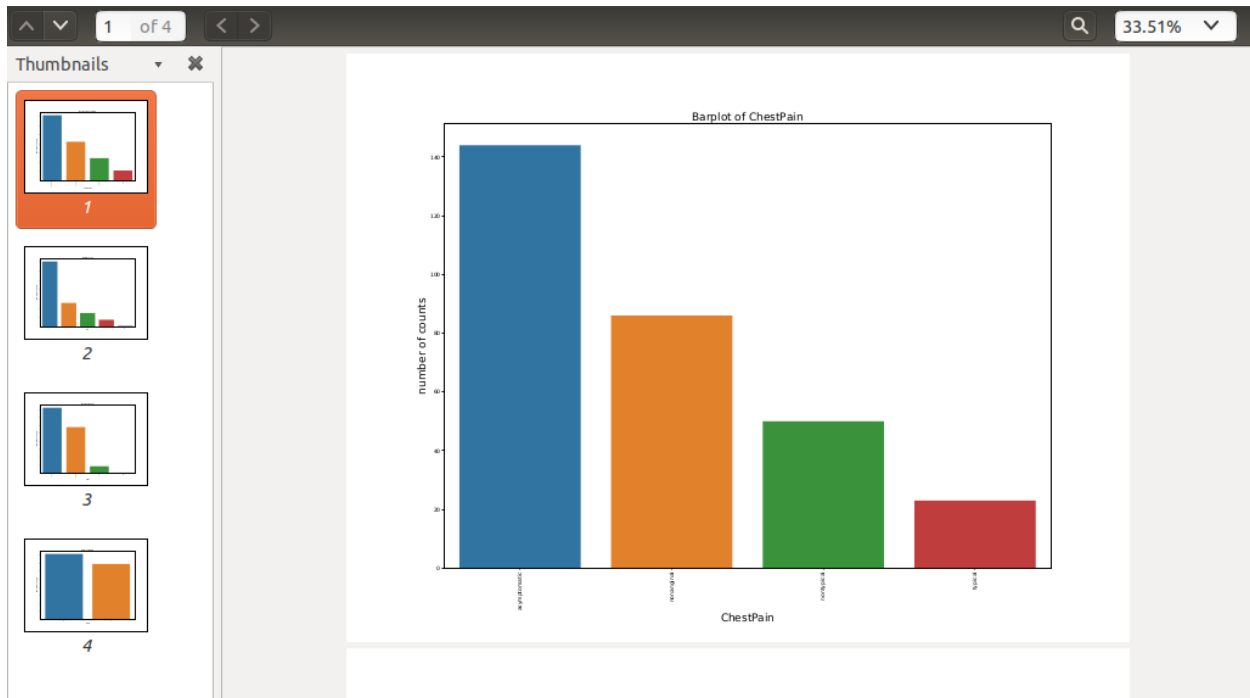
```

from pyspark.sql import SparkSession

spark = SparkSession \
    .builder \

```

(continues on next page)



(continued from previous page)

```
.appName("Python Spark regression example") \
.config("spark.some.config.option", "some-value") \
.getOrCreate()

# from PySparkAudit import dtypes_class, hist_plot, bar_plot, freq_
→items, feature_len
# from PySparkAudit import dataset_summary, rates, trend_plot

# path = '/home/feng/Desktop'

# import PySpark Audit function
from PySparkAudit import auditing

# load dataset
data = spark.read.csv(path='Heart.csv',
                      sep=',', encoding='UTF-8', comment=None,
→header=True, inferSchema=True)

# auditing in one function
print(auditing(data, display=True))
```

Result:

4.2.1 print in bash

```

=====
The audited results summary audited_results.xlsx was located at:
/home/feng/Dropbox/MyTutorial/PySparkAudit/test/Audited
=====
The correlation matrix plot Corr.png was located at:
/home/feng/Dropbox/MyTutorial/PySparkAudit/test/Audited
=====
The Histograms plot Histograms.pdf was located at:
/home/feng/Dropbox/MyTutorial/PySparkAudit/test/Audited
Histograms plots are done!
=====
The Bar plot Bar_plots.pdf was located at:
/home/feng/Dropbox/MyTutorial/PySparkAudit/test/Audited
Caution: no date features in the dataset!!!
Generate all audited results took = 29.093122243881226 s
=====
The auditing processes are DONE!!!
(   feature  dtypes  row_count    ...    rate_neg  rate_zero  rate_pos
0      Age      int      303      ...      0.0      0.000000  1.000000
1       Sex      int      303      ...      0.0      0.320132  0.679868
2    RestBP      int      303      ...      0.0      0.000000  1.000000
3       Chol      int      303      ...      0.0      0.000000  1.000000
4        Fbs      int      303      ...      0.0      0.851485  0.148515
5    RestECG      int      303      ...      0.0      0.498350  0.501650
6      MaxHR      int      303      ...      0.0      0.000000  1.000000
7     ExAng      int      303      ...      0.0      0.673267  0.326733
8   Oldpeak  double      303      ...      0.0      0.326733  0.673267
9      Slope      int      303      ...      0.0      0.000000  1.000000

[10 rows x 22 columns],      feature  dtypes    ...    rate_null  _
→rate_empty
0  ChestPain  string    ...      0.0      0.0
1         Ca  string    ...      0.0      0.0
2        Thal  string    ...      0.0      0.0
3         AHD  string    ...      0.0      0.0

[4 rows x 12 columns],      Age      Sex      RestBP    ...    _
→      ExAng  Oldpeak  Slope
Age      1.000000 -0.097542  0.284946    ...      0.091661  0.203805  0.
→161770
Sex      -0.097542  1.000000 -0.064456    ...      0.146201  0.102173  0.
→037533
RestBP   0.284946 -0.064456  1.000000    ...      0.064762  0.189171  0.
→117382

```

(continues on next page)

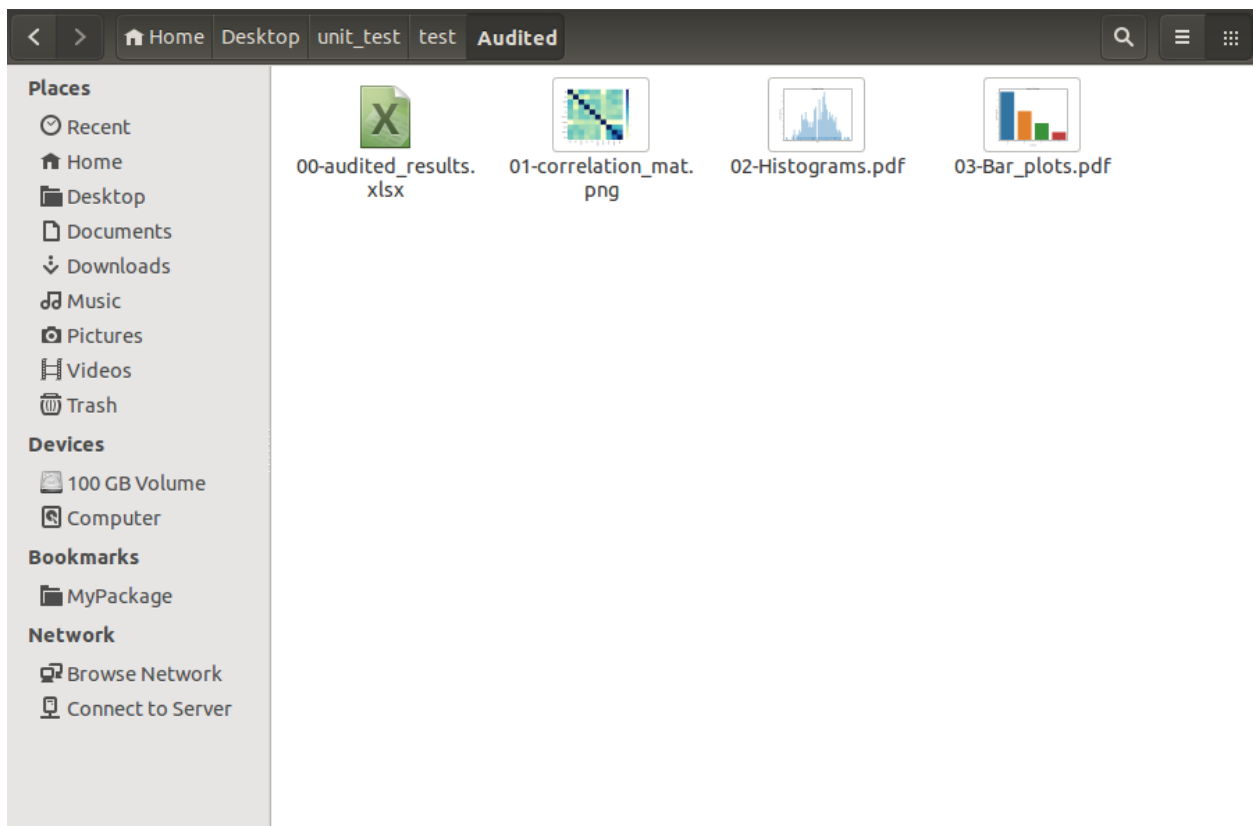
(continued from previous page)

```
Chol      0.208950 -0.199915  0.130120  ...    0.061310  0.046564 -0.
↪004062
Fbs       0.118530  0.047862  0.175340  ...    0.025665  0.005747  0.
↪059894
RestECG   0.148868  0.021647  0.146560  ...    0.084867  0.114133  0.
↪133946
MaxHR     -0.393806 -0.048663 -0.045351 ...   -0.378103 -0.343085 -0.
↪385601
ExAng     0.091661  0.146201  0.064762  ...    1.000000  0.288223  0.
↪257748
Oldpeak   0.203805  0.102173  0.189171  ...    0.288223  1.000000  0.
↪577537
Slope     0.161770  0.037533  0.117382  ...    0.257748  0.577537  1.
↪000000
```

```
[10 rows x 10 columns])
```

Process finished with `exit` code 0

4.2.2 Audited results folder



The files in `00-audited_results.xlsx`:

1. Dataset_summary

	A	B	C	D	E	F	G	H	I	J
1	summary	value								
2	sample_size	303								
3	feature_size	14								
4	single_unique_feature	0								
5	row_w_null	0								
6	row_w_empty	0								
7	row_w_zero	299								
8	row_avg_null_count	0								
9	row_avg_empty_count	0								
10	row_avg_zero_count	3.25083								
11	numerical_fields	10								
12	categorical_fields	4								
13	date_fields	0								
14	unsupported_fields	0								
15	double	1								
16	int	9								
17	string	4								
18										
19										
20										
21										
22										
23										
24										
25										
26										

2. Numeric_summary

3. Category_summary

4. Correlation_matrix

5. Histograms in Histograms.pdf

Calibri											
A1											
	A	B	C	D	E	F	G	H	I	J	K
1	feature	dtypes	row_count	null_count	distinct_count	mean	stddev	min	max	Q1	Med
2	Age	int	303	303	41	54.4389	9.03866	29	77	48	56
3	Sex	int	303	303	2	0.679867	0.467298	0	1	0	1
4	RestBP	int	303	303	50	131.6897	17.59974	94	200	120	130
5	Chol	int	303	303	152	246.6930	51.77691	126	564	211	242
6	Fbs	int	303	303	2	0.148514	0.356197	0	1	0	0
7	RestECG	int	303	303	3	0.990099	0.994971	0	2	0	1
8	MaxHR	int	303	303	91	149.6072	22.87500	71	202	134	153
9	ExAng	int	303	303	2	0.326732	0.469794	0	1	0	0
10	Oldpeak	double	303	303	40	1.039603	1.161075	0.0	6.2	0	0.8
11	Slope	int	303	303	3	1.600660	0.616226	1	3	1	2
12											
13											
14											
15											
16											
17											
18											
19											
20											
21											
22											
23											
24											
25											
26											

Dataset_summary Numeric_summary Category_summary Correlation_matrix

Find Find All Match Case

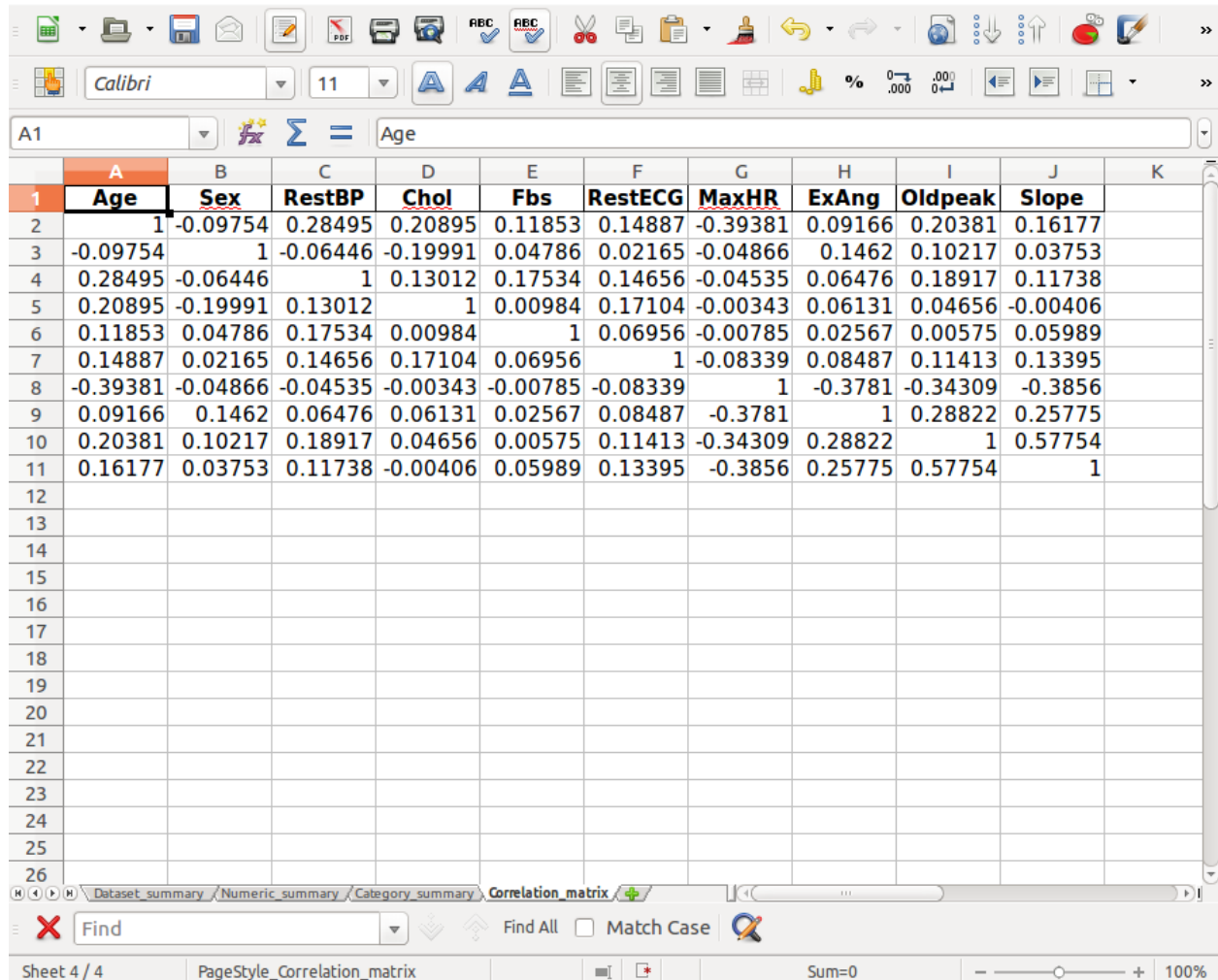
Sheet 2 / 4 PageStyle_Numeric_summary Sum=0 100%

	A	B	C	D	E	F	G	H	I	J	K
1	feature	dtypes	row_count	null_count	distinct_count	min_length	avg_length	max_length	ms[valu	ure_vari	rate_nul
2	ChestPain	string	303	303	4	7	10.7228	12	[[['asympt	0.0132	0
3	Ca	string	303	303	5	1	1.0132	2	[[['0', 176	0.0165	0
4	Thal	string	303	303	4	2	7.45875	10	[[['normal	0.0132	0
5	AHD	string	303	303	2	2	2.45875	3	[[['No', 16	0.0066	0
6											
7											
8											
9											
10											
11											
12											
13											
14											
15											
16											
17											
18											
19											
20											
21											
22											
23											
24											
25											
26											

Dataset_summary Numeric_summary **Category_summary** Correlation_matrix

Find Find All Match Case

Sheet 3 / 4 PageStyle_Category_summary Sum=0 100%

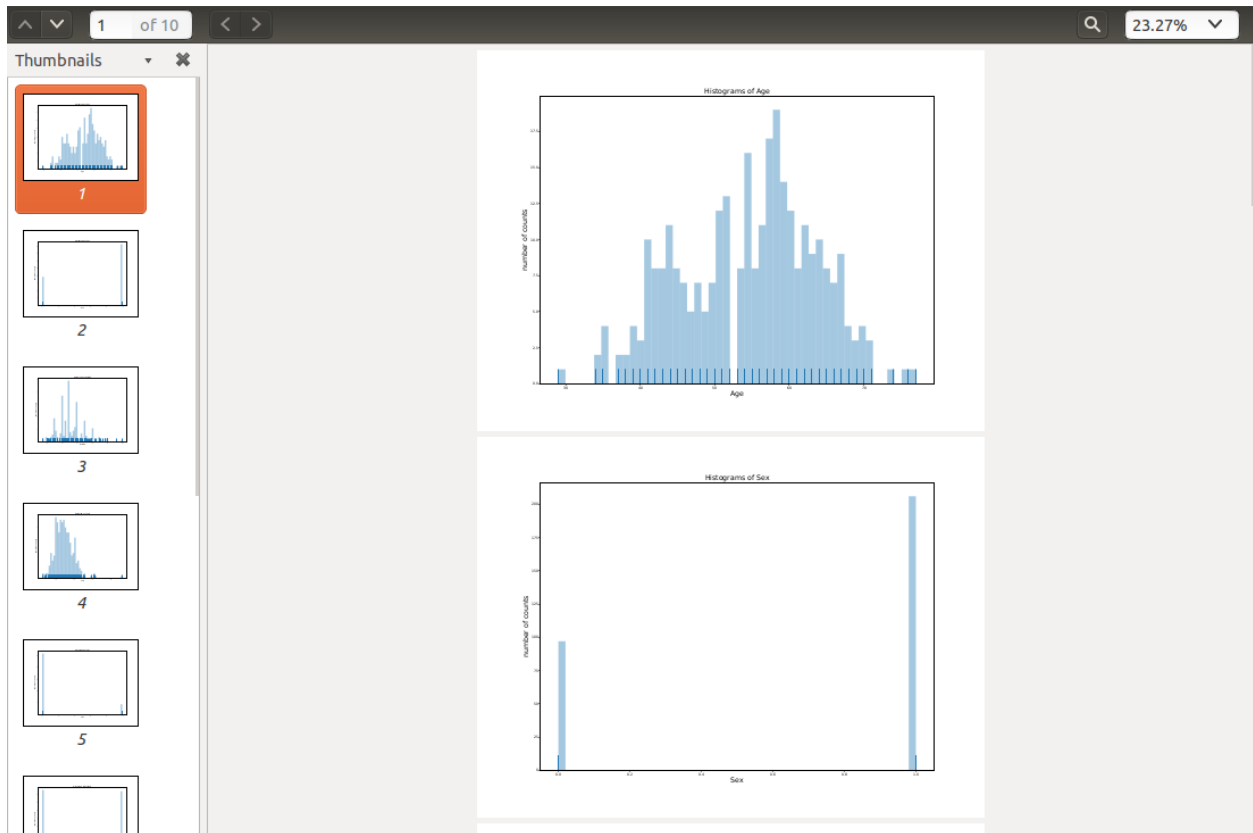


	A	B	C	D	E	F	G	H	I	J	K
1	Age	Sex	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	
2	1	-0.09754	0.28495	0.20895	0.11853	0.14887	-0.39381	0.09166	0.20381	0.16177	
3	-0.09754	1	-0.06446	-0.19991	0.04786	0.02165	-0.04866	0.1462	0.10217	0.03753	
4	0.28495	-0.06446	1	0.13012	0.17534	0.14656	-0.04535	0.06476	0.18917	0.11738	
5	0.20895	-0.19991	0.13012	1	0.00984	0.17104	-0.00343	0.06131	0.04656	-0.00406	
6	0.11853	0.04786	0.17534	0.00984	1	0.06956	-0.00785	0.02567	0.00575	0.05989	
7	0.14887	0.02165	0.14656	0.17104	0.06956	1	-0.08339	0.08487	0.11413	0.13395	
8	-0.39381	-0.04866	-0.04535	-0.00343	-0.00785	-0.08339	1	-0.3781	-0.34309	-0.3856	
9	0.09166	0.1462	0.06476	0.06131	0.02567	0.08487	-0.3781	1	0.28822	0.25775	
10	0.20381	0.10217	0.18917	0.04656	0.00575	0.11413	-0.34309	0.28822	1	0.57754	
11	0.16177	0.03753	0.11738	-0.00406	0.05989	0.13395	-0.3856	0.25775	0.57754	1	
12											
13											
14											
15											
16											
17											
18											
19											
20											
21											
22											
23											
24											
25											
26											

Dataset_summary / Numeric_summary / Category_summary / Correlation_matrix

Find Find All ☐ Match Case

Sheet 4 / 4 PageStyle_Correlation_matrix Sum=0 100%



6. Barplots in Bar_plots.pdf

4.3 Auditing on Big Dataset

In this section, we will demonstrate the auditing performance and audited results on the big data set. The data set is Spanish High Speed Rail tickets pricing. It is available at : <https://www.kaggle.com/thegurus/spanish-high-speed-rail-system-ticket-pricing>. This data set has 2579771 samples and 10 features.

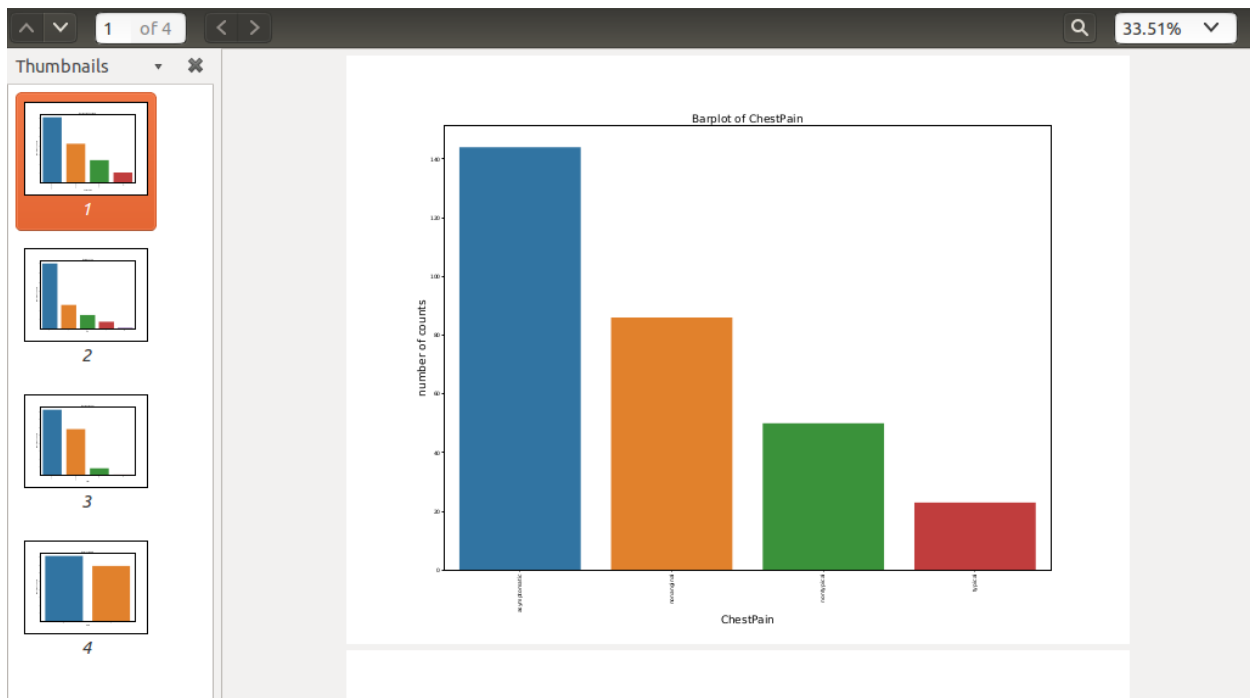
From the following CPU time, you will see most of the time was spent on plotting the histograms. If your time and memory are limited, we will suggest you to use `sample_size` to generate the subset of the the dataset to plot histograms.

For example:

```
from pyspark.sql import SparkSession

spark = SparkSession \
    .builder \
```

(continues on next page)



(continued from previous page)

```
.appName("Python Spark regression example") \
.config("spark.some.config.option", "some-value") \
.getOrCreate()

# from PySparkAudit import dtypes_class, hist_plot, bar_plot, freq_
→items, feature_len
# from PySparkAudit import dataset_summary, rates, trend_plot

# Audited results output path
out_path = '/home/feng/Desktop'

# import PySpark Audit function
from PySparkAudit import auditing

# load dataset
# Spanish High Speed Rail tickets pricing - Renfe (~2.58M)
# https://www.kaggle.com/thegurus/spanish-high-speed-rail-system-
→ticket-pricing

data = spark.read.csv(path='/home/feng/Downloads/renfe.csv',
                      sep=',', encoding='UTF-8', comment=None,
→header=True, inferSchema=True)

# auditing in one function
```

(continues on next page)

(continued from previous page)

```
auditing(data, output_dir=out_path, tracking=True)
```

Result:

4.3.1 print in bash

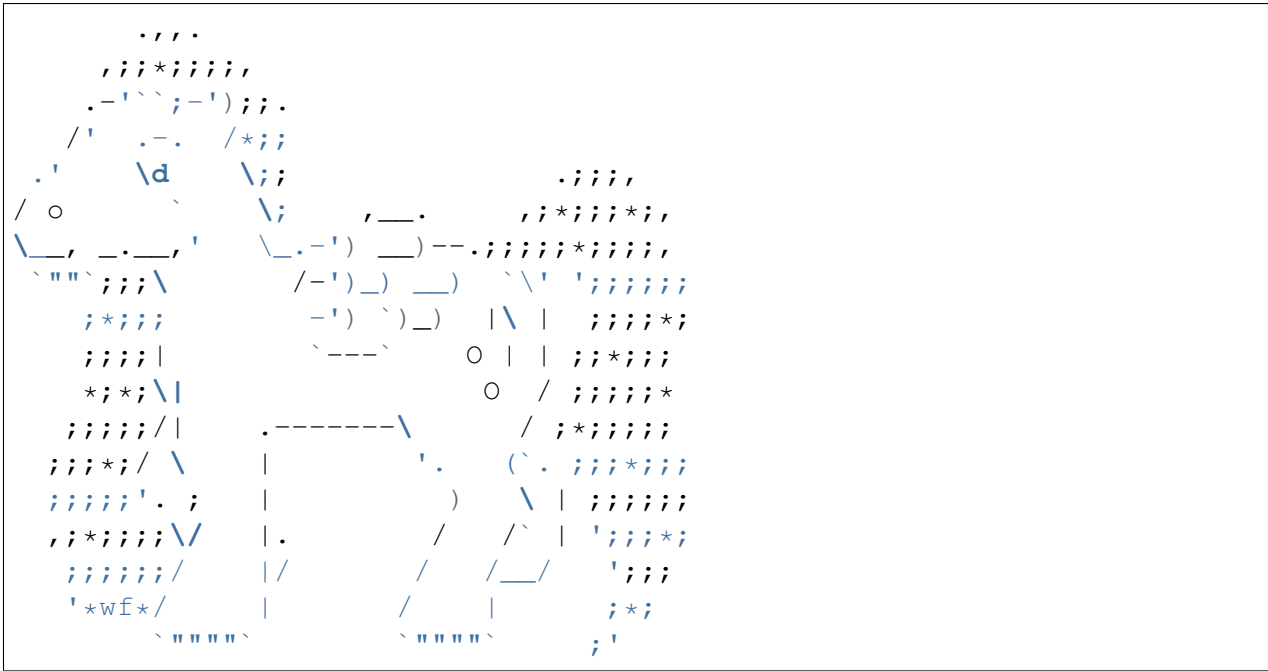
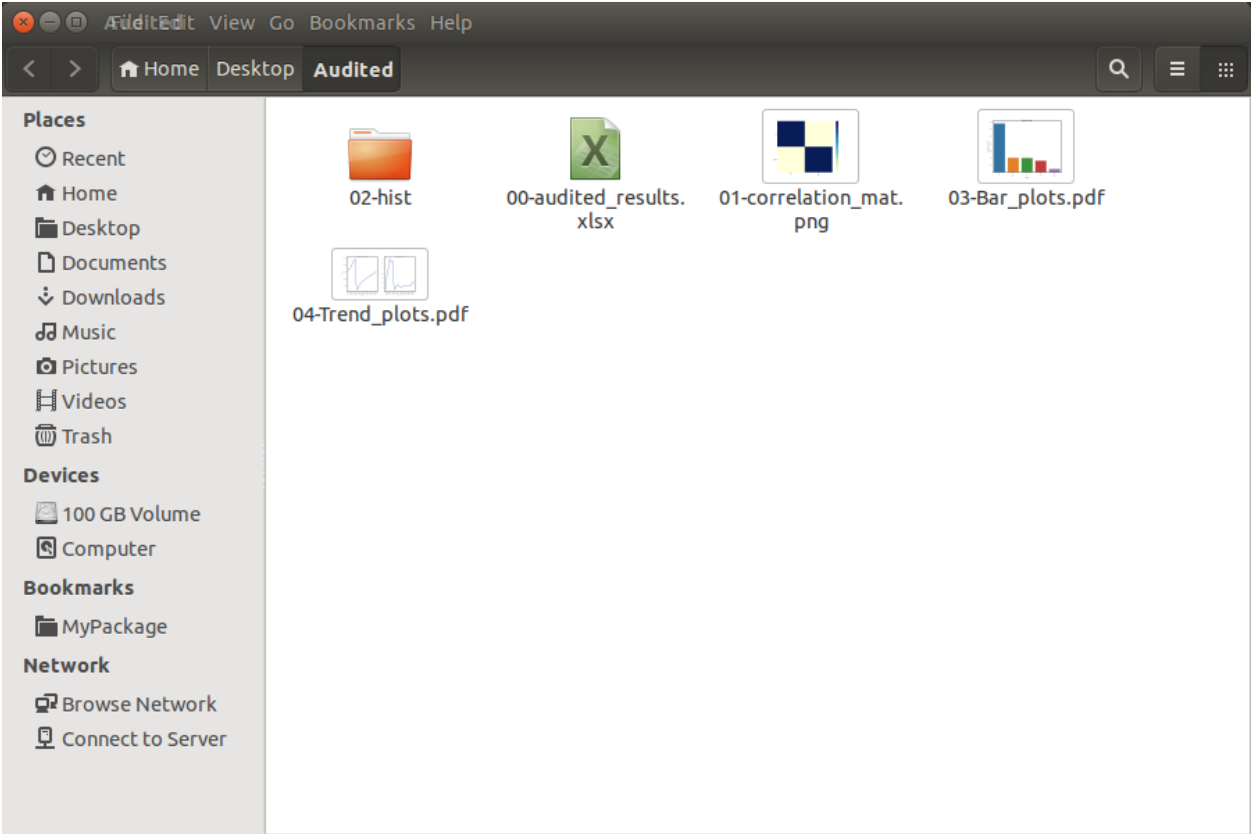
```
=====
The audited results summary audited_results.xlsx was located at:
/home/feng/Desktop/Audited
Generate data set summary took = 60.535009145736694 s
=====
Collecting data types.... Please be patient!
Generate counts took = 0.0016515254974365234 s
=====
Collecting features' counts.... Please be patient!
Generate counts took = 6.502962350845337 s
=====
Collecting data frame description.... Please be patient!
Generate data frame description took = 1.5562639236450195 s
=====
Calculating percentiles.... Please be patient!
Generate percentiles took = 19.76785445213318 s
=====
Calculating features' length.... Please be patient!
Generate features' length took = 4.953453540802002 s
=====
Calculating top 5 frequent items.... Please be patient!
Generate rates took: 4.761325359344482 s
=====
Calculating rates.... Please be patient!
Generate rates took: 17.201056718826294 s
Auditing numerical data took = 54.77840781211853 s
=====
Collecting data types.... Please be patient!
Generate counts took = 0.001623392105102539 s
=====
Collecting features' counts.... Please be patient!
Generate counts took = 12.59226107597351 s
=====
Calculating features' length.... Please be patient!
Generate features' length took = 5.332952976226807 s
=====
Calculating top 5 frequent items.... Please be patient!
Generate rates took: 6.832213878631592 s
=====
```

(continues on next page)

(continued from previous page)

```
=====
Calculating rates.... Please be patient!
Generate rates took: 23.704302072525024 s
Auditing categorical data took = 48.484763622283936 s
=====
The correlation matrix plot Corr.png was located at:
/home/feng/Desktop/Audited
Calculating correlation matrix... Please be patient!
Generate correlation matrix took = 19.61273431777954 s
=====
The Histograms plots *.png were located at:
/home/feng/Desktop/Audited/02-hist
Plotting histograms of _c0.... Please be patient!
Plotting histograms of price.... Please be patient!
Histograms plots are DONE!!!
Generate histograms plots took = 160.3421311378479 s
=====
The Bar plot Bar_plots.pdf was located at:
/home/feng/Desktop/Audited
Plotting barplot of origin.... Please be patient!
Plotting barplot of destination.... Please be patient!
Plotting barplot of train_type.... Please be patient!
Plotting barplot of train_class.... Please be patient!
Plotting barplot of fare.... Please be patient!
Plotting barplot of insert_date.... Please be patient!
Plotting barplot of start_date.... Please be patient!
Plotting barplot of end_date.... Please be patient!
Bar plots are DONE!!!
Generate bar plots took = 24.17994236946106 s
=====
The Trend plot Trend_plots.pdf was located at:
/home/feng/Desktop/Audited
Plotting trend plot of _c0.... Please be patient!
Plotting trend plot of price.... Please be patient!
Trend plots are DONE!!!
Generate trend plots took = 11.697550296783447 s
Generate all the figures took = 196.25823402404785 s
Generate all audited results took = 379.73954820632935 s
=====
The auditing processes are DONE!!!
```

4.3.2 Audited results folder



MAIN REFERENCE

BIBLIOGRAPHY

[PyAudit] Wenqiang Feng and Ming Chen. [Python Data Audit Library API](#), 2019.

[PySparkAudit] Wenqiang Feng and Yiming Xu. [PySpark Data Audit Library API](#), 2019.