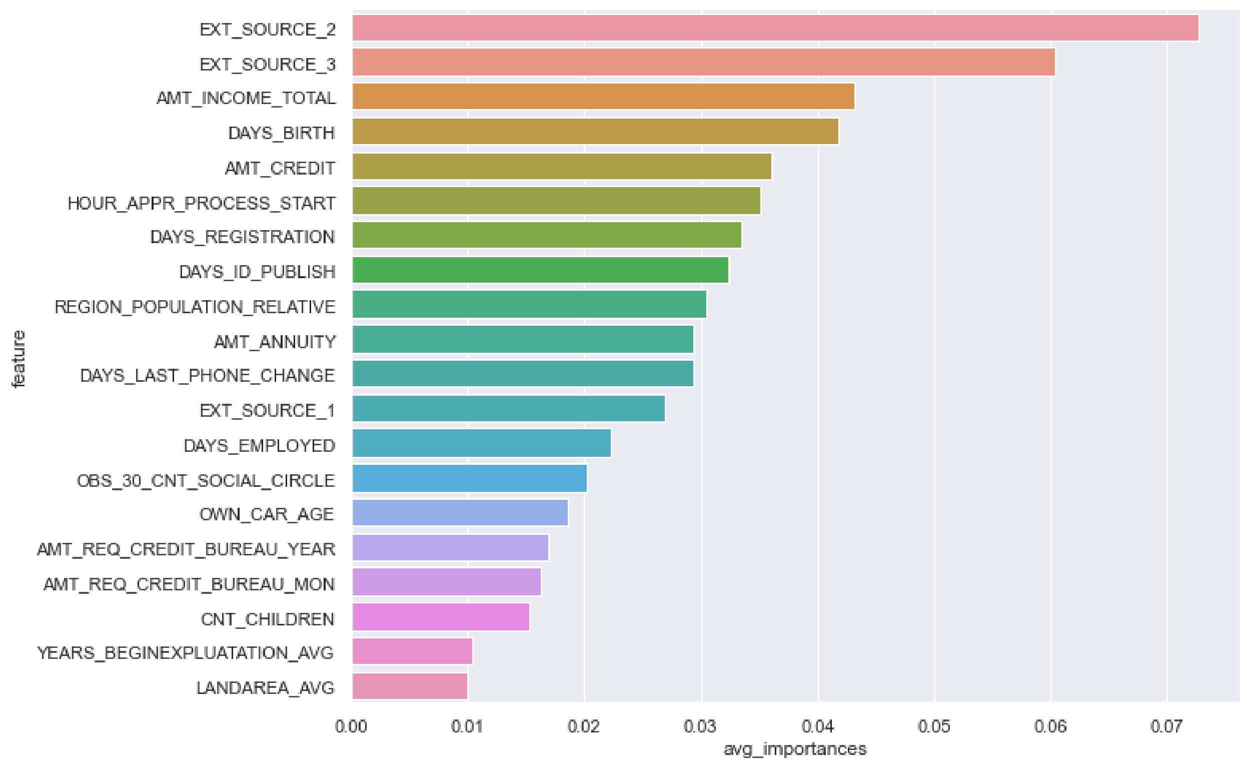


AutoFeatures: PySpark Auto Feature Selector

Release 1.0

Wenqiang Feng

July 07, 2020

CONTENTS

Welcome to our **AutoFeatures: PySpark Auto Feature Selector**!!! The PDF version can be downloaded from HERE.

CHAPTER ONE

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 AutoFeatures: PySpark Auto Feature Selector [AutoFeatures] API. The PDF version can be downloaded from HERE. **You may download and distribute it. Please beaware, 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.

1.1.2 About the author

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

- **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 and DST.

1.2 Feedback and suggestions

Your comments and suggestions are highly appreciated. I am more than happy to receive corrections, suggestions or feedback through email (Wenqiang Feng: von198@gmail.com) for improvements.

CHAPTER

TWO

HOW TO INSTALL

2.1 Install with `pip`

You can install the `PyAudit` from [PyPI](https://pypi.org/project/AutoFeatures):

```
pip install AutoFeatures
```

2.2 Install from Repo

2.2.1 Clone the Repository

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

2.2.2 Install

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

2.2.3 Uninstall

```
pip uninstall AutoFeatures
```

2.2.4 Test

CHAPTER THREE

AUTOFEATURES CLASS

3.1 Utils Functions

3.1.1 Data types

class AutoFeatures.**AutoFeatures**

Auto feature selector for Machine Learning Modeling with PySpark. This class has four selectors:

1. unique selector: identify the single unique features
2. missing selector: identify missing values features with missing threshold
3. collinear selector: identify collinear features with threshold
4. low importance selector: identify low importance features with Gradient Boosting Machine(GBM)

classmethod 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
```

(continues on next page)

(continued from previous page)

```
>>> 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'], [])
```

classmethod get_dummy(*df_in*, *index_col=None*, *categorical_cols=None*, *continuous_cols=None*, *label_col=None*, *dropLast=False*)

Get dummy variables and concat with continuous variables for ml modeling.

Parameters

- **df_in** – the dataframe
- **categorical_cols** – the name list of the categorical data
- **continuous_cols** – the name list of the numerical data
- **label_col** – the name of label column
- **dropLast** – the flag of drop last column

Returns encoded dummy variable names and feature matrix

Author Wenqiang Feng

Email von198@gmail.com

```
>>> df = spark.createDataFrame([
              (0, "a"),
              (1, "b"),
              (2, "c"),
              (3, "a"),
              (4, "a"),
              (5, "c")
          ], ["id", "category"])
```

```
>>> index_col = 'id'
>>> categorical_cols = ['category']
>>> continuous_cols = []
>>> label_col = []
```

```
>>> mat = get_dummy(df,index_col,categorical_cols,continuous_cols,
↪label_col)
>>> mat.show()
```

```
>>>
    +---+-------------+
    | id|     features|
    +---+-------------+
    |  0|[1.0,0.0,0.0]|
    |  1|[0.0,0.0,1.0]|
    |  2|[0.0,1.0,0.0]|
    |  3|[1.0,0.0,0.0]|
    |  4|[1.0,0.0,0.0]|
    |  5|[0.0,1.0,0.0]|
    +---+-------------+
```

classmethod get_encoded_names(*df_in*, *categorical_cols*)
get the encoded dummy variable names

Parameters

- **df_in** – the input dataframe
- **categorical_cols** – the name list of the categorical columns

Returns the name list of the encoded dummy variable for categorical columns

3.2 AutoFeatures Class

class AutoFeatures.**AutoFeatures**
Auto feature selector for Machine Learning Modeling with PySpark. This class has four selectors:

1. unique selector: identify the single unique features
2. missing selector: identify missing values features with missing threshold
3. collinear selector: identify collinear features with threshold
4. low importance selector: identify low importance features with Gradient Boosting Machine(GBM)

__init__()
Initialize self. See help(type(self)) for accurate signature.

classmethod corr_selector(*data*, *index_col=None*, *label_col=None*, *corr_thold=0.9*, *method='pearson'*, *rotation=True*, *display=False*, *tracking=False*, *cat_num=2*)
collinear selector: identify collinear features with threshold

Parameters

- **data** – input dataframe
- **index_col** – the name of the index column and the other columns you want to exclude
- **label_col** – the name of the label column
- **corr_thold** – threshold for collinear scores

- **`method`** – the method to use for computing correlation, supported: pearson (default), spearman
- **`rotation`** – the flag of rotate x-ticks
- **`display`** – the flag for displaying plots, the default value is False
- **`tracking`** – the flag for displaying CPU time, the default value is False
- **`cat_num`** – the number of the categorical feature (helping removing binary features)

Returns The name list of the correlated values features above threshold

`classmethod ensemble_drop`(*data*, *index_col*, *label_col*, *task*, *importance_thold=None*, *cumulative_thold=0.96*, *missing_thold=0.6*, *corr_thold=0.9*, *method='pearson'*, *rotation=True*, *n_train=5*, *top_n=20*, *dropLast=False*, *display=False*, *tracking=False*, *cat_num=2*)

Ensemble drop (based on essential drop, that is to say it has included the functionals of essential drop) is a method to identify the essential drop features based on ensemble ML model (GBM).

Parameters

- **`data`** – input dataframe
- **`index_col`** – the name of the index column and the other columns you want to exclude
- **`label_col`** – the name of the label column
- **`task`** – the ensemble model type, supported task "classification" or "regression"
- **`importance_thold`** – the threshold of the feature importance if missing will be auto calculated by the cumulative threshold
- **`cumulative_thold`** – the threshold of the cumulative feature importance, this will be used to determine the importance_thold when importance_thold is missing
- **`missing_thold`** – threshold for missing values percentage
- **`corr_thold`** – threshold for collinear scores
- **`method`** – the method to use for computing correlation, supported: pearson (default), spearman
- **`rotation`** – the flag of rotate x-ticks
- **`n_train`** – the numbers of train for average the feature importance
- **`top_n`** – the numbers for plot top_n highest feature importance
- **`dropLast`** – the flag of the drop last column during applying the OneHotEncoder
- **`display`** – the flag for displaying plots, the default value is False

- **tracking** – the number of the categorical feature (helping removing binary features)
- **cat_num** – the number of the categorical feature (helping removing binary features)

Returns The name list of the to_drop features with low feature importance

classmethod essential_drop(*data*, *index_col*, *label_col*, *missing_thold=0.6*, *corr_thold=0.9*, *method='pearson'*, *rotation=True*, *display=False*, *tracking=False*, *cat_num=2*)

Essential drop (included: missing selector, unique selector, correlation selector) is all in one functions to identify the essential drop features.

Parameters

- **data** – input dataframe
- **index_col** – the name of the index column and the other columns you want to exclude
- **label_col** – the name of the label column
- **missing_thold** – threshold for missing values percentage
- **corr_thold** – threshold for collinear scores
- **method** – the method to use for computing correlation, supported: pearson (default), spearman
- **rotation** – the flag of rotate x-ticks
- **display** – the flag for displaying plots, the default value is False
- **tracking** – the flag for displaying CPU time, the default value is False
- **cat_num** – the number of the categorical feature (helping removing binary features)

Returns The name list of the to_drop features with essential drop functions

classmethod importance_selector(*data*, *index_col*, *label_col*, *task*, *importance_thold=None*, *cumulative_thold=0.96*, *missing_thold=0.6*, *corr_thold=0.9*, *method='pearson'*, *rotation=True*, *n_train=5*, *top_n=20*, *dropLast=False*, *display=False*, *tracking=False*, *cat_num=2*)

importance selector: identify low feature importance features with threshold

Parameters

- **data** – input dataframe
- **index_col** – the name of the index column and the other columns you want to exclude
- **label_col** – the name of the label column

- **`task`** – the ensemble model type, supported task "classification" or "regression"
- **`importance_thold`** – the threshold of the feature importance if missing will be auto calculated by the cumulative threshold
- **`cumulative_thold`** – the threshold of the cumulative feature importance, this will be used to determine the importance_thold when importance_thold is missing
- **`missing_thold`** – threshold for missing values percentage
- **`corr_thold`** – threshold for collinear scores
- **`method`** – the method to use for computing correlation, supported: pearson (default), spearman
- **`rotation`** – the flag of rotate x-ticks
- **`n_train`** – the numbers of train for average the feature importance
- **`top_n`** – the numbers for plot top_n highest feature importance
- **`dropLast`** – the flag of the drop last column during applying the OneHotEncoder
- **`display`** – the flag for displaying plots, the default value is False
- **`tracking`** – the number of the categorical feature (helping removing binary features)
- **`cat_num`** – the number of the categorical feature (helping removing binary features)

Returns The name list of the dropped features with low feature importance

`classmethod missing_selector`(*data*, *missing_thold=0.6*, *display=False*, *tracking=False*)

Missing selector: identify missing values features with missing threshold

Parameters

- **`data`** – input dataframe
- **`missing_thold`** – threshold for missing values percentage
- **`display`** – the flag for displaying plots, the default value is False
- **`tracking`** – the flag for displaying CPU time, the default value is False

Returns The name list of the missing values features above missing threshold

`classmethod unique_selector`(*data*, *tracking=False*)

Unique selector: identify the single unique features

Parameters

- **`data`** – input dataframe
- **`tracking`** – the flag for displaying CPU time, the default value is False

Return unique_drop The name list of the single unique features

CHAPTER

FOUR

AUTOFEATURES DEMOS

The following demos are designed to show how to use `AutoFeatures` to select proper features.

4.1 AutoFeatures Essential Drop

For example:

```
# simple test
from AutoFeatures import AutoFeatures


from pyspark.sql import SparkSession

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

my_list = [('a', 2, 3),
           ('b', 5, 6),
           ('c', 8, 9),
           ('a', 2, 3),
           ('b', 5, 6),
           ('c', 8, 9)]
col_name = ['col1', 'col2', 'col3']

df = spark.createDataFrame(my_list, schema=col_name)



df.show()

Fs = AutoFeatures()
indexCol = []
labelCol = []

to_drop = Fs.essential_drop(df, index_col=indexCol, label_col=labelCol,␣
↪missing_thold=0.68, corr_thold=0.9,
```

(continues on next page)

(continued from previous page)

```
                            method="pearson", rotation=True, display=True,␣
↪tracking=True, cat_num=2)

print('essential dropped features:{}'.format(to_drop))
```

Result:

```
+----+----+----+
|col1|col2|col3|
+----+----+----+
|   a|   2|   3|
|   b|   5|   6|
|   c|   8|   9|
|   a|   2|   3|
|   b|   5|   6|
|   c|   8|   9|
+----+----+----+

Unique selector took = 6.319664716720581 s
Missing selector took = 17.472286224365234 s
Correlation selector took = 28.78574252128601 s
The essential selector took = 65.23012638092041 s
essential dropped features:['col3']
```

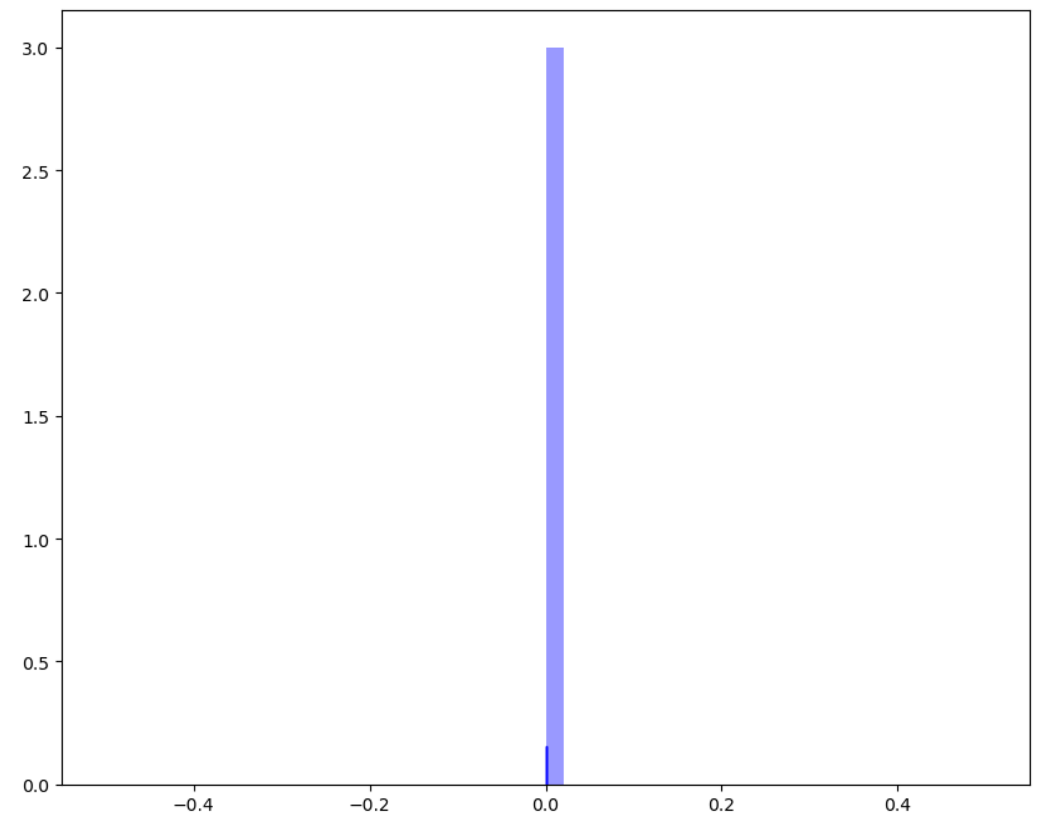
4.2 AutoFeatures Ensemble Drop

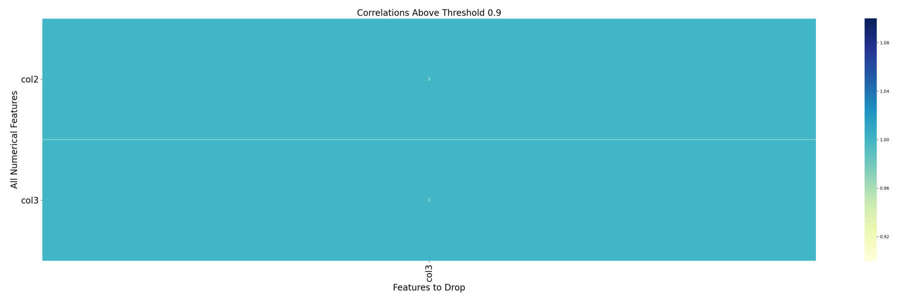
4.2.1 Classification

For example:

```
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'

from AutoFeatures import AutoFeatures

# load dataset
```

(continues on next page)


3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.4
−0.2
0.0
0.2
0.4


Correlations Above Threshold 0.9
col2
col3
All Numerical Features
col3
Features to Drop

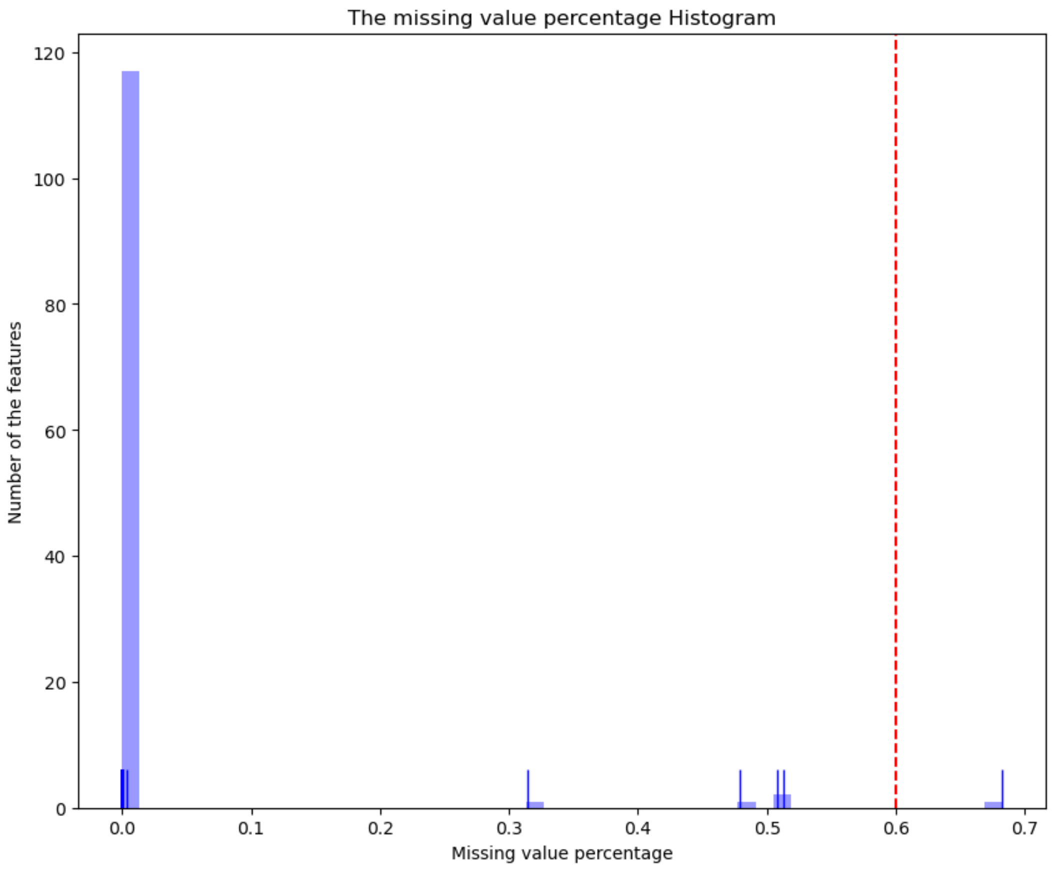
(continued from previous page)

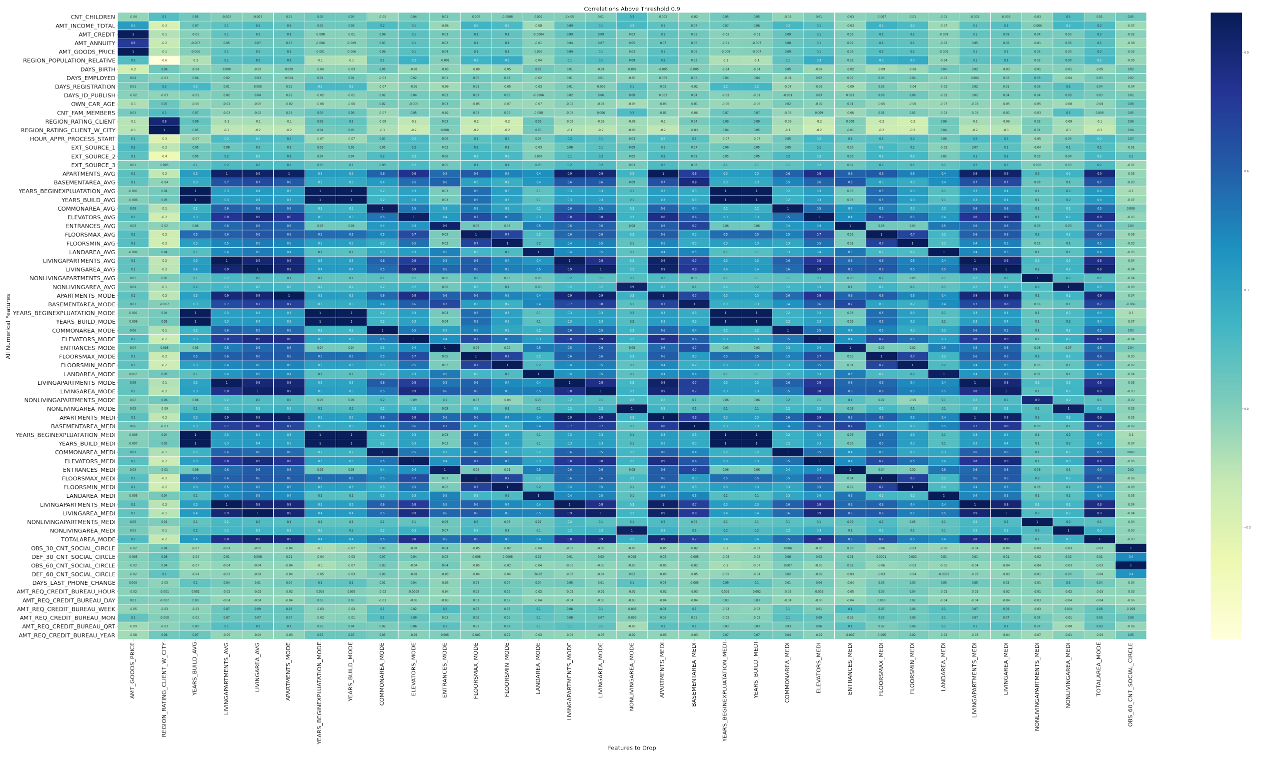
```
data = spark.read.csv(path='../data/credit_example.csv',
                      sep=',', encoding='UTF-8', comment=None, header=True,␣
↪inferSchema=True)
data = data.fillna(0)

print(data.toPandas().head(5))

indexCol = ['SK_ID_CURR']
labelCol = 'TARGET'

task = 'classification'

Fs = AutoFeatures()

# correlation selector
to_drop = Fs.corr_selector(data, index_col=indexCol, label_col=labelCol,
                           corr_thold=0.9, method="pearson", rotation=True,
                           display=False, tracking=False, cat_num=2)
print('corr_selector::{}'.format(to_drop))

# essential selector (included: missing selector, unique selector,␣
↪correlation selector)
to_drop = Fs.essential_drop(data, index_col=indexCol, label_col=labelCol,
                            missing_thold=0.6, corr_thold=0.9, method="pearson
↪", rotation=True,
                            display=True, tracking=True, cat_num=2)
print('essential_drop::{}'.format(to_drop))

# ensemble selector (ensemble selector is based on essential selector.)
to_drop = Fs.ensemble_drop(data, index_col=indexCol, label_col=labelCol,␣
↪task=task, tracking=True)
print('ensemble_drop::{}'.format(to_drop))
```

Result:

```
   SK_ID_CURR  TARGET  ... AMT_REQ_CREDIT_BUREAU_QRT AMT_REQ_CREDIT_BUREAU_
↪YEAR
0      247408       0  ...                       0.0                        1.
↪0
1      153916       0  ...                       0.0                        0.
↪0
2      229065       0  ...                       0.0                        7.
↪0
3      282013       0  ...                       0.0                        1.
↪0
4      142266       0  ...                       1.0                        1.
↪0

[5 rows x 122 columns]
```

and


The missing value percentage Histogram
Number of the features
Missing value percentage
0
20
40
60
80
100
120
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7



4.2.2 Regression

```
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'

from AutoFeatures import AutoFeatures

# load dataset
data = spark.read.csv(path='../data/credit_example.csv',
                      sep=',', encoding='UTF-8', comment=None, header=True,␣
↪inferSchema=True)
data = data.fillna(0)
```

(continues on next page)

(continued from previous page)

```
print(data.toPandas().head(5))

indexCol = ['SK_ID_CURR', 'CODE_GENDER']
labelCol = 'AMT_INCOME_TOTAL'

task = 'regression'

Fs = AutoFeatures()


# essential selectors (included: missing selector, unique selector,␣
↪correlation selector)
to_drop = Fs.essential_drop(data, index_col=indexCol, label_col=labelCol,
                            missing_thold=0.68, corr_thold=0.9, method=
↪"pearson", rotation=True,
                            display=True, tracking=True, cat_num=2)
print('essential_drop:')
print(to_drop)

# ensemble selectors

to_drop = Fs.ensemble_drop(data, index_col=indexCol, label_col=labelCol,␣
↪task=task)
print('ensemble_drop:')
print(to_drop)
```

CHAPTER

FIVE

MAIN REFERENCE

BIBLIOGRAPHY

[AutoFeatures] Wenqiang Feng and Ming Chen. Python Data Audit Library API, 2019.

INDEX