DATA SCIENCE & ENGINEERING SUMMER INTERNSHIP

AN AUTO-MACHINE-LEARINING FRAMEWORK BASED ON PYTHON AND TENSORFLOW

Richard Xia

10 August 2020

HSBC

SECTION 1

PROJECT OVERVIEW

Looking at key summaries as well as demonstrating how project helps HSBC DSE team

ACCELERATING YOUR BIG DATA WORKFLOW

Project aims to provide a centralized, integrated, universal, cross-lab and pip-installable toolkit for data pre-processing and an automated framework for model construction and optimization covering all types of tasks.

INTEGRATED PYTHON SOLUTION

All functions achieved with two objects:

1. **Data Object** covering preprocessing
2. **Model object** for auto-hyper parameter tuning

ADVANCED DATA CLEANING TECHNICS

Deep learning based missing number imputation
MDLP algorithm for variable discretization
Entity Embedding for network-based vectorization

AUTO FEATURE SELECTION

Automatic feature selection process by stating number of feature needed:
RF importance | LASSO | Chi-2 | LightGBM

DIAGNOSTIC TOOL BUILD-IN

Tools provided to check:

1. Data quality score
2. Missing value summary report
3. Visualization

MIXED VARIABLES HANDLING

Handling both categorical and numerical variable
Automatically detect feature type and pass to suitable algorithm.
FAMD | TFIDF | MFA | Entity Embedding

EASY HYPER-PARAMETER TUNING

Hyper-parameter of the networks can be tuned automatically, and the optimal model structures will be detected without too much effort.

PROJECT STRUCTURE

DataObject and HyperModel

PROJECT STRUCTURE

DataFrame Object

Integrated Data Pre-Processing Toolkits

A — Data Quality Diagnostic

Correlation Visualization, Data Type Report, Missing value report

B — Missing Number Imputation

Drop | mean | median | mode | KNN | GAIN | | MICE | datawig

C — Data Scale Transformation

Outlier removal and scale transformation

D — Dimension Reduction

PCA | MFA | FAMD for both numerical and categorical entry

E — Encoder & Discretization

Embedding of categorical value
Discretization of numerical vcalue

F — Automatic Feature Selection

Chi-2 | RFE | LASSO | Random Forest | Gradient Boosting

HyperModel Object

Automatic Hyper-Parameter Tuning Framework

1. Hyperband hyper-parameter tuning method
2. Random search
3. Bayesian optimisation

PROJECT RATIONAL
How the project is helping the business

Data Science & Engineering Team

Three different pods
- Business Development Pod
- Engineering Pod
- Delivery Pod

Improving Data Asset
Data factory covers function ingestion, data privacy, data engineering, data tagging, entity resolution

Productize ML project
Leverage Data asset to increase revenue and cut cost, including reducing risk weighted asset

Generate new business insight
Project cross over to global banking market and GLCM. And we helps make data driven decision to better serve our clients.

Business Development Pod

BDP aims to generate insight on corporate client data, idea generation, R&D as well as model innovation.

STEP 1
Idea generation
Proof of Concept

Looking for insights from huge amount of data assets. And validate the conceptual possibility both analytically and technically across different teams.

1

STEP 2
Data Analysis
Model construction

If the concept is proven working, then pod need to work on EDA for observing the patterns. The pod will also work on model development and cross validation.

2

STEP 3
Stakeholder engagement
Implementation

If the model was working as expected, then pod needs to communicate with relevant parties to demonstrate model and to improve on new business expectations.

3

Multiple projects going on same time

Limited code sharing across projects

Relatively new DSE team

Project is able to provide an integrated toolkit for data pre-processing and an automated framework for model construction

CONSTRUCT

A Model Asset
An Analysis Toolkit
A Code Repository

A **Standardizing and templating data pre-processing**
Templating advanced data analysis algorithms for instant implementation in different project across team

B **Integrating code from different project**
Model functionalities can be enlarged if people contribute part of algorithms from individual projects into the system

C **Achieving higher efficiency**
Allow more time on insight generation and model validation instead of writing code from scratch again and again

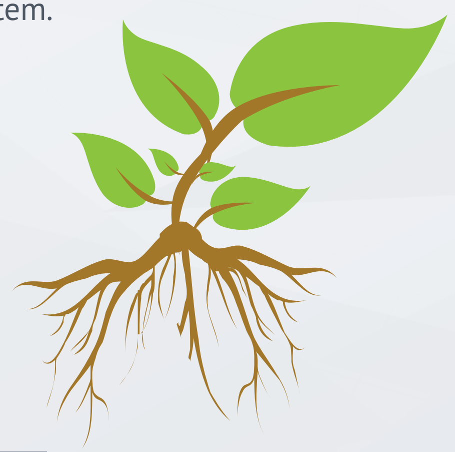
D **An automated hyperparameter tuning framework**
support exploring all possible model structure with minimum amount coding required

ULTIMATE GOAL OF MY PROJECT

Developed an integrated coding solution across different teams

HSBC data science team are working with so many different projects, but there are limited active sharing of codes or algorithms, even though many of the tasks are highly similar, like EDA, imputation and category encoding.

I hope ultimate form of this project is to successfully create a centralised, integrated, regularly-expanded, pip-installable toolkit and framework that is accessible to everyone, to all labs in the DSE team, which will reduce the time spent on repetitive works of coding, and allow more time for idea generation and business analysis. And an integrated toolkit can let everyone be able to easily use the more task-specific algorithms regardless the level of exposure, accelerate their learning curve. In addition, the DSE team can constantly expand the versatility of framework if people contribute part of useful algorithms in their past projects into the system.



Finished

STAGE **ONE**

Integrated pre-processing and parameter tuning framework for binary classification problem, and already implemented in various projects.

-

STAGE **TWO**

- More time-series data pre-processing methods
- Include regression and timeseries prediction option into model tuning framework

-

STAGE **THREE**

People across teams contributes together some reusable codes in the past projects in order to expand as well as to create a mature centralised toolkit for the whole group

HSBC

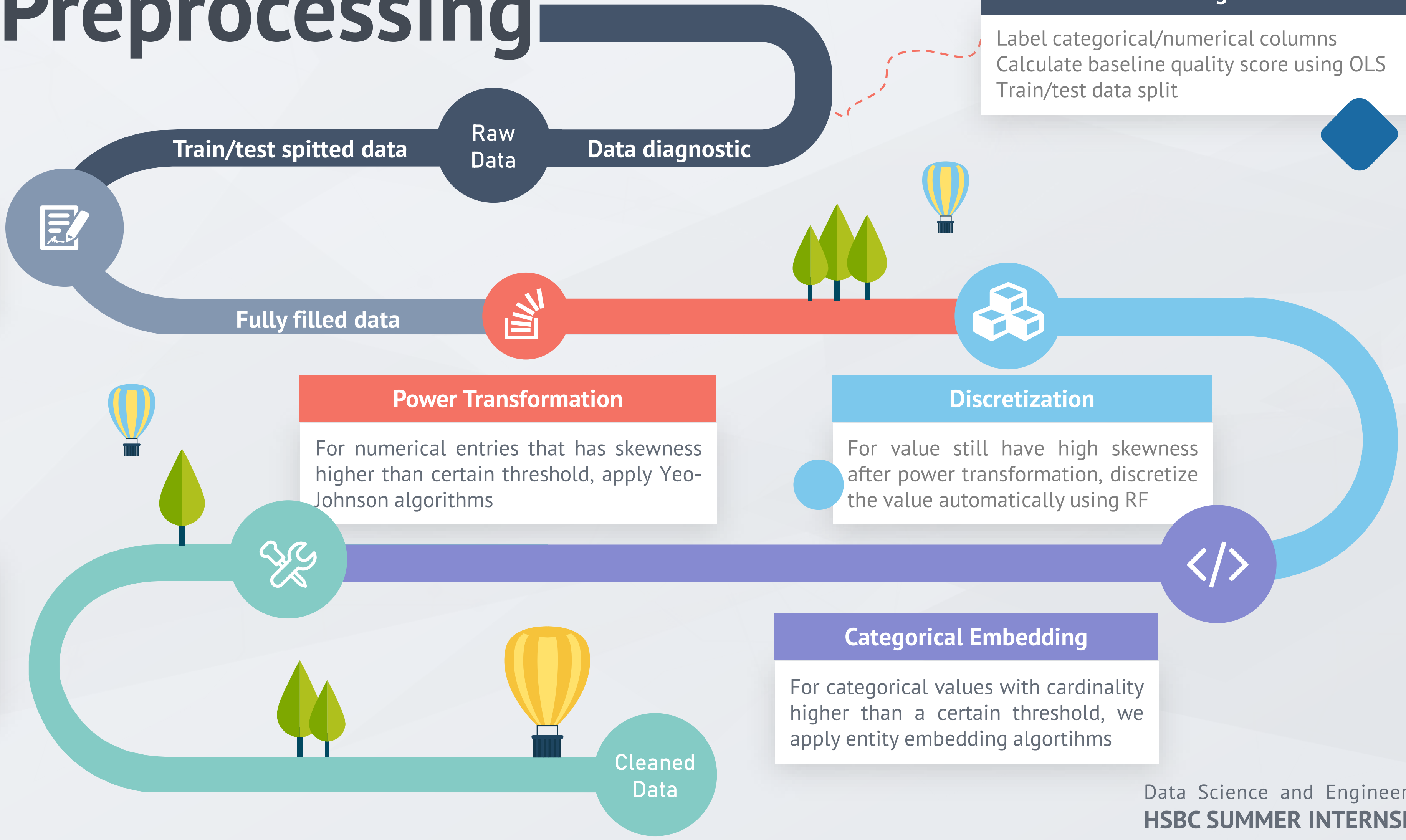
SECTION 2

DATAFRAME OBJECT

Looking at how this data object overs the FULL life-cycle of data pre-processing and cleaning

Integrated DataFrame object roadmap for

Automatic Preprocessing



Data diagnostic

Label categorical/numerical columns
Calculate baseline quality score using OLS
Train/test data split

Missing value imputation

Numerical columns fill missing by KNN algorithms. Categorical columns filled by datawig or max_occurrence

Power Transformation

For numerical entries that has skewness higher than certain threshold, apply Yeo-Johnson algorithms

Discretization

For value still have high skewness after power transformation, discretize the value automatically using RF

Normalization

We standardize all entries including the embedded categorical values of the dataset before passing into the hypermodel object for auto hyper-parameter tuning.

Categorical Embedding

For categorical values with cardinality higher than a certain threshold, we apply entity embedding algortihms

How SIMPLE could it be?

Because of the limitation of using actual HSBC second order risk project data, here I used a highly similar personal loan dataset on predicting whether a people will default after the loan being approved.

The data has 33 features with 157000 numbers of personal load record.

Data Exploration

Constructor: credit_data = Non_TimeSeries_DataFrame(df = pd.read_csv('credit.csv'), label_col='def_flag')

Actions it takes: Train/Test data split | Categorical/Numerical/hybrid features detection | Baseline data quality score | number string converter

	def_flag	int_rate	emp_length_p	home_ ownership	Annual_inc	verification_ status	purpose_p	term	revol_bal	delinq_2yrs	...
0	False	12.99	10.0	MORTGAGE	63000.0	Not Verified	debt_ consolidation	36	8453	1	...
1	True	14.16	2.0	RENT	156000.0	Verified	debt_ consolidation	36	12625	2	...
2	False	8.39	5.0	OWN	21000.0	Source Verified	home_ improvement	36	6364	0	...
3	False	16.99	10.0	RENT	162000.0	Source Verified	credit_card	60	11923	0	...

Check missing values: credit_data.na_report

	Variable	Missing Values	Filling Factor (%)
0	emp_length_p	8045	94.878569
1	avg_cur_bal	2	99.998727
2	def_flag	0	100.000000
3	mo_sin_old_rev_tl_op	0	100.000000

Check data format: credit_data.data_type

	Column	Type	Sample_text	Unique_value
0	def_flag	categorical	False	2.0
9	term	categorical	36	2.0
11	loan_amnt	numerical	4000	NaN
32	addr_state	categorical	NJ	47.0

How SIMPLE could it be?

Data Exploration → Imputation → Outlier → Power Transform → Dim Reduction → Discretize → Embedding → Done

Imputation:	credit_data.imputate({ 'MICE' : ['emp_length_p' , 'avg_cur_bal'], 'MaxOccurance' : ['addr_state'] }
Numerical outlier removal:	credit_data.outlier_removal(columns = data.num_col, limits = [0.05,0.95], method = 'Quantiles')
Power Transform:	credit_data.scale_transformation(columns = [], method = 'Yeo-Johnson', auto_para = 7) Here columns are automatically selected from numerical data if its absolute skewness is greater than 7, but you can also overwrite it
Dimension Reduction:	credit_data.auto_merge_high_correlation(columns = [] , threshold = 0.75) Here columns are automatically merged if the correlation are higher than the threshold. It support numerical and categorical features
Discretization:	credit_data.discretization(columns = [], method = 'DecisionTree') Here columns are automatically selected if numerical value is still skewed after power transformation. (This can also be applied on integer category feature, such as risk grade and account types, i.e. using value 0,1,2,3,4…. only)
Embedding:	credit_data.feature_encoder(columns = [], method = 'EntityEmbedding', target_dimension = 4) Here columns are automatically selected if cordiality of the feature is greater than 10.

Now the data is ready for passing to classification model

PART 2: INTEGRATED DATA OBJECT

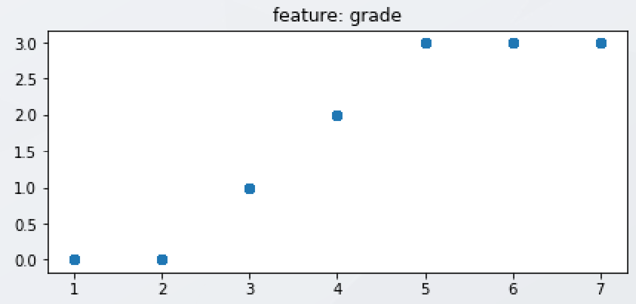
Covering FULL life-cycle of data pre-processing

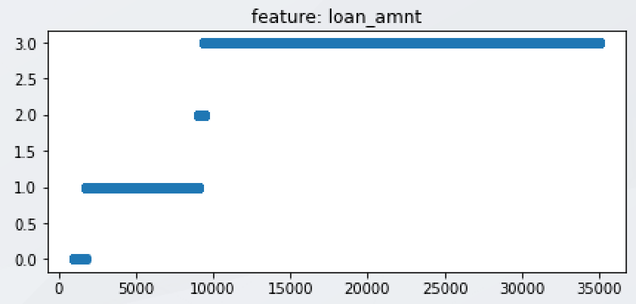
Example 1

Second order risk project with Xueyan
Applying discretization algorithm on several features of the dataset such as risk grade and borrowing limits.

Applying the discretization algorithm will:

1. Reduce the noise in the dataset and improve signal-to-noise ratio
2. Fitting a model to bins reduces the impact that small fluctuates in the data
3. Each bin "smooths" out the fluctuates/noises in sections of the data.
4. Achieve better interpretability. Low risk client, middle risk client and high risk clients rather than risk level 1,2,3… Small business, middle size borrow and VIP clients instead of size of borrowing limits

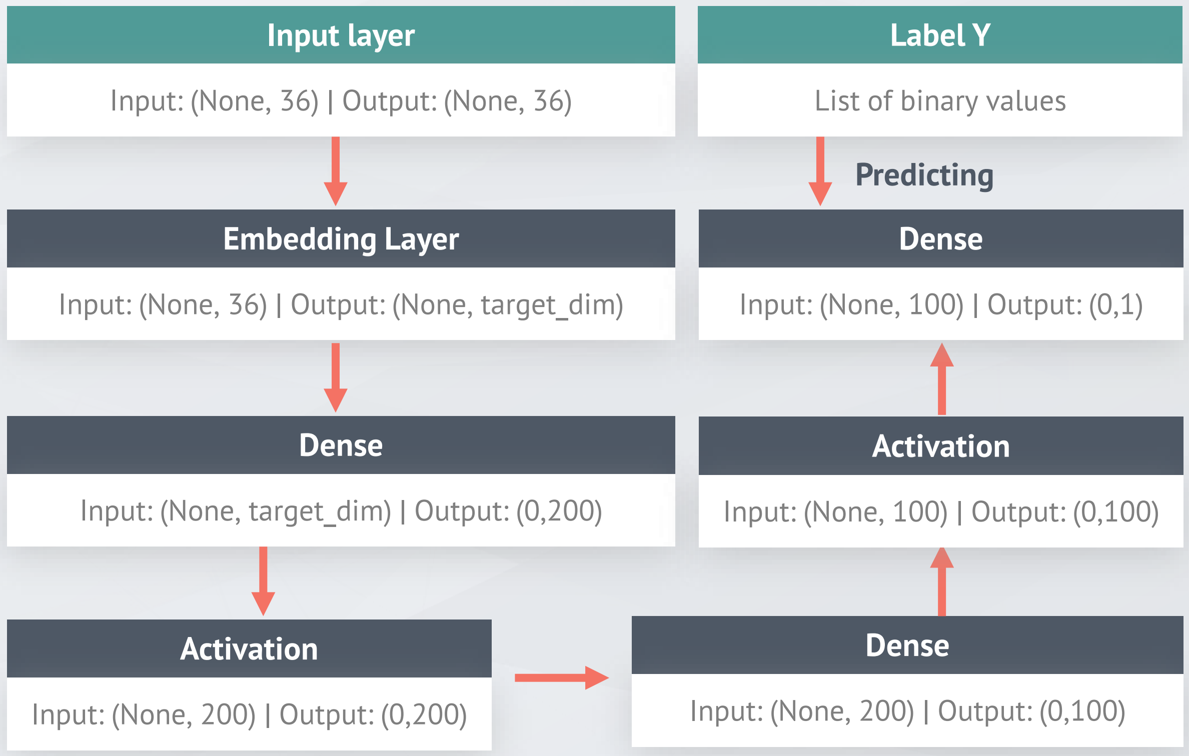




Example 2

Entity Embedding on project with Victoria.

- One-hot encoding of high cardinality features often results in an unrealistic amount of computational resource requirement and generating large amount of dummy columns, which may intensify the curse of dimensionality.
- It treats different values of categorical variables completely independent of each other and often ignores the informative relations between them.


Input layer
Input: (None, 36) | Output: (None, 36)
Embedding Layer
Input: (None, 36) | Output: (None, target_dim)
Dense
Input: (None, target_dim) | Output: (0,200)
Activation
Input: (None, 200) | Output: (0,200)
Dense
Input: (None, 200) | Output: (0,100)
Activation
Input: (None, 100) | Output: (0,100)
Dense
Input: (None, 100) | Output: (0,1)
Predicting
Label Y
List of binary values

HSBC

SECTION 2 Additional

DATAFRAME OBJECT

Detailed introduction to technical details of the model and visualization of effectiveness of each method

Part 2.1:

Data Quality Diagnostic

This aims to reduce the time spending on decision make process on deciding which pre-process steps need to be taken and what types of technics to be applied in next few steps.

Correlation Visualization | **Missing Value Report** | **Data Type Check**

In a normal workflow, people need to check the correlation of features first to avoid singularity during the matrix transformation. Also need to check the amount of missing values to decide what type of missing value imputation algorithms to be used. Finally, a data type check can figure out whether numbers are saved as numeric and how many unique values do a categorical feature have.

Data Type Report

It will show the **machine detected columns type**, **the example row** and **number of unique values** available

It the column type is categorical, but sample text is some numbers, then it means the number are saved as string. If the column is categorical, but the number of unique values are too large, then we need some embedding algorithm for the categorical features

	Column	Type	Sample_text	Unique_value
0	def_flag	categorical	False	2.0
11	loan_amnt	numerical	4000	NaN
32	addr_state	categorical	NJ	47.0

Missing value report

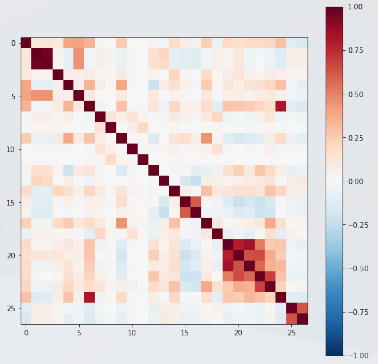
It will show the feature names, the number of missing values as well as what proportion of entries are filled. For example, in the table below, we know we need to imputate the columns of 'emp_length_p' and 'avg_cur_bal'. And since only small amount of entry is missing, an imputation by mean value is enough.

	Variable	Missing Values	Filling Factor (%)
0	emp_length_p	8045	94.878569
1	avg_cur_bal	2	99.998727
2	def_flag	0	100.000000
3	mo_sin_old_rev_tl_op	0	100.000000

Correlation Report

It will draw the correlation matrix to let user understanding the correlation between different features visually. And it will also report pairs of highest correlation features to alert the user.

Column 1	Column 2	Correlation
int_rate	grade	0.967237
num_actv_bc_tl	num_bc_sats	0.837267
revol_bal	total_rev_hi_lim	0.813094
num_actv_bc_tl	num_actv_rev_tl	0.784332



PART 2.2:

Missing Value Imputation

This aims to improve the efficiency of imputation process by integrating advanced imputation method into the codes and reducing the time spending to looking for solution online and compiling the codes.

Drop	Mean/Median/Mode	Max Occurrence
KNN	GAIN	MICE

Missing number imputation is important because when number of missing value is big, we could not simply use mean method to fill in the missing value. And we cannot simply remove the entries with the missing value as we will waste a lot of useful information.

Method	Type	Explanation
drop	Both	Remove the rows that contain the missing value
mean	numerical only	Interpolate missing value using the mean value
median	numerical only	Interpolate missing value using the median value
mode	numerical only	Interpolate missing value using the mode value
max_occurance	categorical only	Impute value based on most frequently occurred category
KNN	numerical only	Look at k most similar rows and impute way by weighted average
GAIN	numerical only	Generative adversarial imputation networks (gain)
Simple_Datawig	Both	Network-based imputation using datawig package
Simple_MICE	numerical only	Multivariate Imputation by Chained Equations

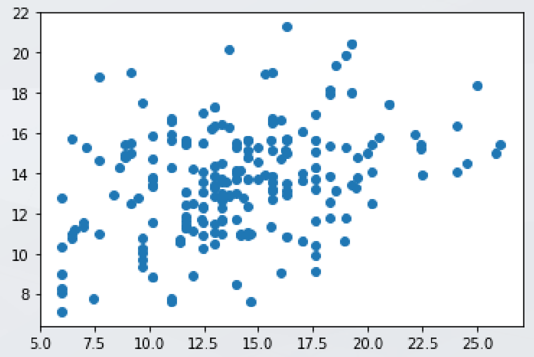
Useful Resources if interested

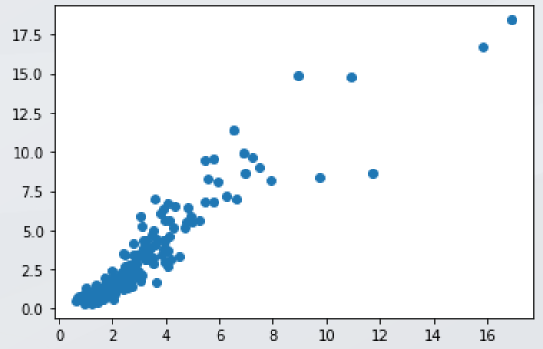
Generative Adversarial Networks: 2017 paper https://arxiv.org/pdf/1708.06724.pdf

DataWig: deep learning based missing value imputation for both numerical and categorical data https://github.com/awslabs/datawig

Multivariate Imputation by Chained Equation (MICE): This type of imputation works by filling the missing data multiple times.
https://www.jstatsoft.org/article/view/v045i03/v45i03.pdf

Example of Imputation Result





The graphs on the above shows a sample imputation with KNN algorithm. The image on the left is working on dataset of personal loan data. The image on the right is working with NYC Green cab data. For the green cab project and the dataset, if you are interested, pls see my github repo: https://github.com/aureole222/NYC_green_cab

The x-axis stands for the actual data and y-axis shows the value imputed. You could see that the performance is fairly okay.

PART 2.3:

Scale Transformation

This aims to improve the accuracy and speed of convergence of neuron network models, as ANN and dimension reduction algorithms such as PCA and FAMD are sensitive to data standardization. It consist of 4 different functions:

Normalization | Power transform | Outlier removal | Outlier removal

In a normal workflow, people need to remove the outliers in the dataset to avoid overfitting. People need to scale the numeric values in case it has very long tails (such as personal income). Using the object build-in function can greatly reduce the time needed in writing the code.

Normalization

Because different features do not have similar ranges of values and hence gradients may end up taking a long time and can oscillate back and forth and take a long time before it can finally find its way to the global/local minimum. To overcome the model learning problem, we normalize the data. We make sure that the different features take on similar ranges of values so that gradient descents can converge more quickly.

Method	Type	Explanation
MaxMin	numerical only	Mean normalization suggests centering the variable at 0 and re-scaling the variable's value range to the range of -1 to 1.
Std	numerical only	Standardization suggests centering the variable at 0 and standardizing the variance to 1.
Robust	numerical only	In this method, the median is used instead of the mean. We remove the median from the variable observations, and then we scale to the inter-quantile range (IQR).
MaxAbs	numerical only	Maximum absolute scaling scales the variable values between -1 and 1 by dividing the data by its maximum value

Power Transformation

Many machine learning algorithms perform better when the distribution of variables is Gaussian. A power transform will make the probability distribution of a variable more Gaussian. This is often described as removing a skew in the distribution, although more generally is described as stabilizing the variance of the distribution. Log transformation is a simplified version of power transformation.

Method	Type	Explanation
Yeo-Johnson	all numeric	This transformation is somewhat of an adjustment to the Box-Cox transformation, by which we can apply it to negative numbers.
Box-Cox	positive only	It's an evolution of the exponential transformation, which looks through various exponents instead of trying them manually

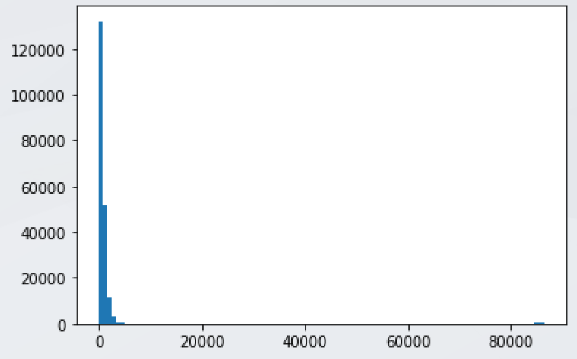
Box-Cox

$$x_i^{(\lambda)} = \begin{cases} \dfrac{x_i^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0, \\ \ln(x_i) & \text{if } \lambda = 0, \end{cases}$$

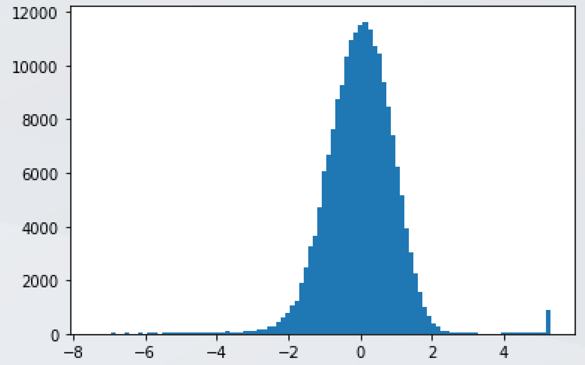
Yeo-Johnson

$$x_i^{(\lambda)} = \begin{cases} [(x_i+1)^\lambda - 1]/\lambda & \text{if } \lambda \neq 0, x_i \geq 0, \\ \ln(x_i) + 1 & \text{if } \lambda = 0, x_i \geq 0 \\ -[(-x_i+1)^{2-\lambda} - 1]/(2-\lambda) & \text{if } \lambda \neq 2, x_i < 0, \\ -\ln(-x_i+1) & \text{if } \lambda = 2, x_i < 0 \end{cases}$$

Before Yeo-Johnson



After Yeo-Johnson



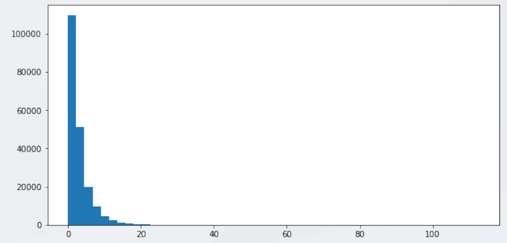
PART 2.3:
Scale Transformation

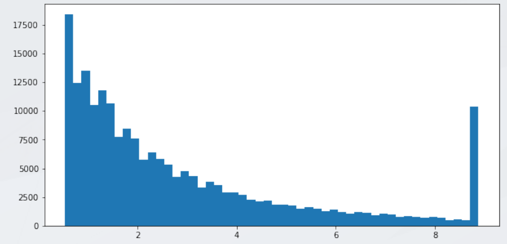
Outlier Removal

Because different features do not have similar ranges of values and hence gradients may end up taking a long time and can oscillate back and forth and take a long time before it can finally find its way to the global/local minimum. To overcome the model learning problem, we normalize the data. We make sure that the different features take on similar ranges of values so that gradient descents can converge more quickly.

Method	Type	Explanation
Trim	numerical only	Trimming merely means removing outliers from the dataset
Inter-quantal	numerical only	The boundaries are determined using IQR proximity rules. Values bigger or smaller than the chosen value are replaced by this value.
Gaussian	numerical only	Boundaries decided by the mean and standard deviation
Quantiles	numerical only	Maximum absolute scaling scales the variable values between -1 and 1 by dividing the data by its maximum value

Before and after using quantiles method

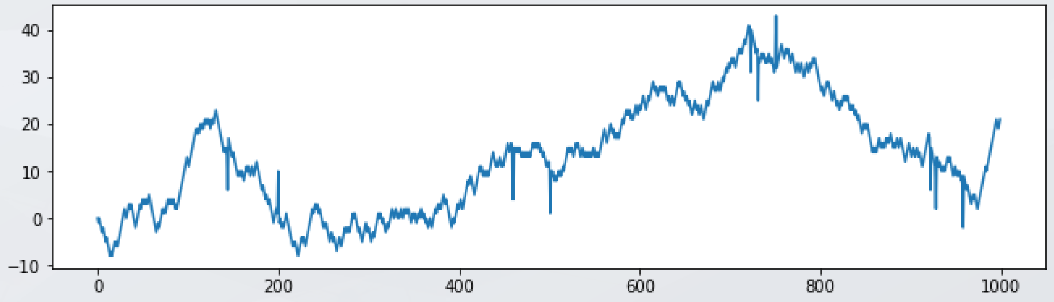


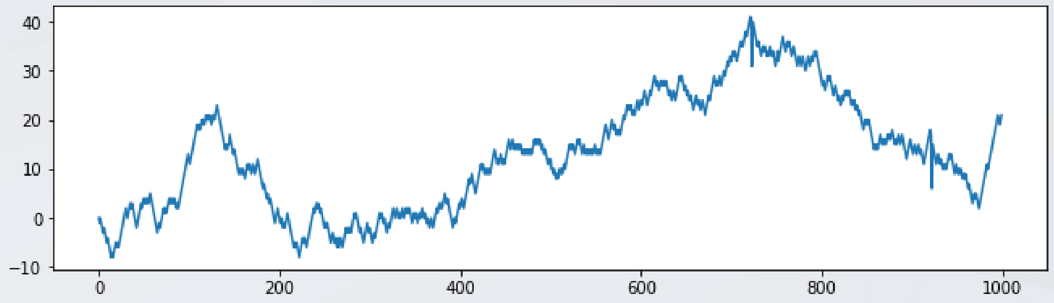


Hampel filter

This function removes outliers **only on time-series object**! The Hampel filter is a member of the class of decision filters that replaces the central value in the data window with the median if it lies far enough from the median to be deemed an outlier. This filter depends on both the window width and an additional tuning parameter n_sigmas. Reducing to the median filter when n_sigmas=0, it may be regarded as another median filter extension.

Before and after using Hampel filter





PART 2.4:

Dimension Reduction

Dimensionality reduction refers to techniques that reduce the number of input variables in a dataset. More input features often make a predictive modelling task more challenging to model, more generally referred to as the curse of dimensionality.

In this project, I used 3 different methods of dimension reduction as well as proposing an automatic approach.

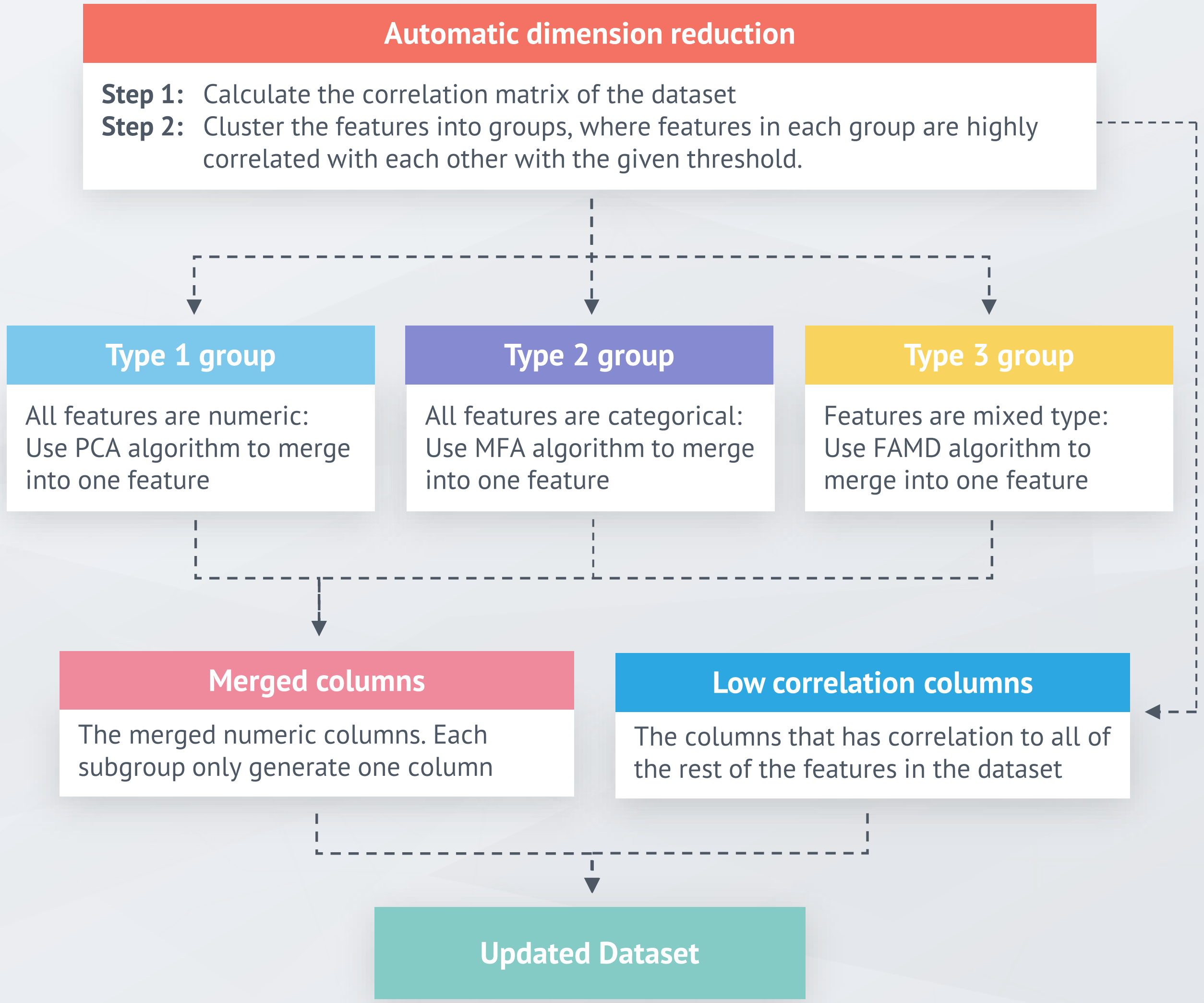
PCA | MFA | FAMD

High-dimensionality statistics and dimensionality reduction techniques are often used for data visualization. Nevertheless these techniques can be used in applied machine learning to simplify a classification or regression dataset in order to better fit a predictive model.

Principle component analysis (PCA)
Principal component analysis (PCA) simplifies the complexity in high-dimensional data while retaining trends and patterns. It does this by transforming the data into fewer dimensions, which act as summaries of features.

Multiple Correspondence analysis (MFA)
The multiple correspondence analysis (MCA), an unsupervised multivariable procedure which can identify non-linear interactions, was used to study the functional connectivity of BG when subjects were at rest. It could convert multiple categorical values into dimension-reduced numerical values.

Factor analysis for mixed data (FAMD)
Factor analysis of mixed data (FAMD) is a principal component method dedicated to analyze a data set containing both quantitative and qualitative variables. It makes it possible to analyze the similarity between individuals by taking into account a mixed types of variables.



Automatic dimension reduction

Step 1: Calculate the correlation matrix of the dataset
Step 2: Cluster the features into groups, where features in each group are highly correlated with each other with the given threshold.

Type 1 group
All features are numeric: Use PCA algorithm to merge into one feature

Type 2 group
All features are categorical: Use MFA algorithm to merge into one feature

Type 3 group
Features are mixed type: Use FAMD algorithm to merge into one feature

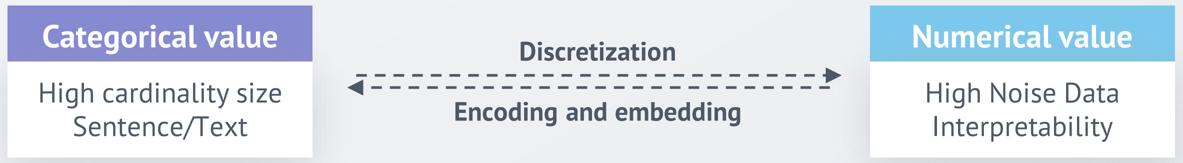
Merged columns
The merged numeric columns. Each subgroup only generate one column

Low correlation columns
The columns that has correlation to all of the rest of the features in the dataset

Updated Dataset

PART 2.5:

Encoder & Discretization



Encoder

We need encode categorical values before we can pass it to machine learning models. For the most of the times, we are simply using one-hot embedding. But it has two main short comings:

- One-hot encoding of high cardinality features often results in an unrealistic amount of computational resource requirement and generating large amount of dummy columns, which may intensify the curse of dimensionality.
- It treats different values of categorical variables completely independent of each other and often ignores the informative relations between them.

So when there are huge amount of unique values in the one categorical variable, or working with text data, one hot encoding may not be the best choice.

Entity embedding maps categorical variables in a function approximation problem into Euclidean spaces, which are the entity embeddings of the categorical variables. The mapping is learned by a neural network during the standard supervised training process.

Method	Type	Explanation
OneHot	numerical	Simple dummy encoding
EntityEmbedding	numerical	Deep learning based multi-dimensional encoding for categorical values
TFIDF	numerical	Encode text data using tfidf word-of-bag representation. Suitable for NLP tasks as well as documents classification

Discretization

One reason to discretize continuous features is to improve signal-to-noise ratio. Fitting a model to bins reduces the impact that small fluctuates in the data has on the model, often small fluctuates are just noise. Each bin "smooths" out the fluctuates/noises in sections of the data. Second reason is for interpretability. Continuous variables such as age are better understood when discretized into meaningful groups: infants, youngsters, young adults, adults, senior, … this is common in the field of marketing

Method	Type	Explanation
EqualWidth	numerical	This is the most simple form of discretization—it divides the range of possible values into N bins of the same width.
EqualFrequency	numerical	Equal-frequency discretization divides the scope of possible values of the variable into N bins, where each bin holds the same number (or approximately the same number) of observations.
Kmeans	numerical	This discretization method consists of applying k-means clustering to the continuous variable—then each cluster is considered as a bin
DecisionTree	numerical	It assigns an observation to one of N end leaves. Accordingly, any decision tree will generate a discrete output, which its values are the predictions at each of its N leaves.
MDLP_Entropy	numerical	Minimum description length principle algorithm to find the optimal number of bins and clusters. If force_multi_iter is true, it will then be combined with information gain discretization algorithm

PART 2.6:

Auto Feature Selection

Feature Selection is a very critical component in a Data Scientist's workflow. When presented data with very high dimensionality, models usually perform badly because:

- **Training time** increases exponentially with number of features.
- Models have increasing risk of **overfitting** with increasing number of features.
- The high **complexity** of a model makes it difficult to interpret.

Feature Selection methods helps with these problems by reducing the dimensions without much loss of the total information. It also helps to make sense of the features and its importance. A simpler model is also easier to interpret and deliver to respective business contact. Here I used 5 different feature selection algorithm:

Chi-2
Random Forest Importance
Recursive Feature Elimation
Gradient Boosting
LASSO Regression / L1 Penalized Logistic Regression

The final selection is made based on majority vote.

An sample project on credit scoring dataset with 150K of records

	Feature	Chi-2	RFE	LASSO	Random Forest	LightGBM	Total
1	mort_acc	True	True	True	True	True	5
2	mo_sin_rcnt_tl	True	True	True	True	True	5
3	mo_sin_rcnt_rev_tl_op	True	True	True	True	True	5
4	mo_sin_old_rev_tl_op	True	True	True	True	True	5
5	loan_amnt	True	True	True	True	True	5
6	int_rate	True	True	True	True	True	5
7	dti	True	True	True	True	True	5
8	annual_inc	True	True	True	True	True	5
9	addr_state	True	True	True	True	True	5
10	total_rev_hi_lim	False	True	True	True	True	4
11	revol_bal	False	True	True	True	True	4
12	num_actv_rev_tl	True	False	True	True	True	4
13	inq_last_6mths	True	True	True	False	True	4
14	grade	True	True	True	False	True	4
15	avg_cur_bal	True	True	True	True	False	4
16	home_ownership	True	True	False	False	True	3

HSBC

SECTION 3

HYPERMODEL OBJECT

Looking at how I set up the auto hyperparameter tuning using Hyperband, Random search and Bayesian optimisation algorithms on sample projects

Data Science and Engineering
HSBC SUMMER INTERNSHIP

HYPERMODEL Object

Hypermodel object is an automated framework that conduct hyperparameter tuning on Tensorflow model, covering all possible parameters in the model construction

- ✓ Hidden Layers
- ✓ Learning Rate
- ✓ Model Structure
- ✓ Optimiser

It gives us the capability to fully customised parameter tuning based on model requirement and task-specific situations.

1 **Initiate the model structure:**
Define the range of hidden layers, nodes type, decay rate, normalization….

2 **Select the optimization metrics**
F1 score | Accuracy | Precession | True/False Positive | Customized Function

3 **Choose the hyperparameter optimizer :**
Hyperband | Random Search | Bayesian Optimisation

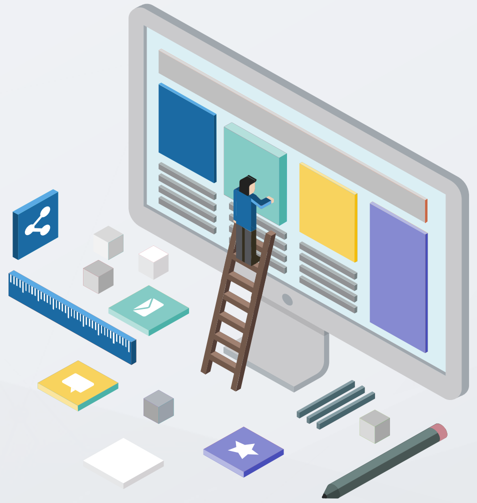
Example 1

```
class MyHyperModel(HyperModel):
  def __init__(self, num_classes):
    self.num_classes = num_classes

  def build(self, hp):
    model = keras.Sequential()
    for i in range(hp.Int('num_layers', 2, 10)):
       model.add(
          keras.layers.Dense(
             units=hp.Int('units_' + str(i), min_value=64,
                                max_value=512, step=32),
             activation='relu’))
       model.add(
          keras.layers.Dropout(
             hp.Float('dropout',min_value=0.0,max_value=0.1,
             default=0.005,step=0.01)))
    model.add(keras.layers.Dense(self.num_classes, activation='sigmoid’))
    model.compile(
       optimizer=keras.optimizers.Adam(
          hp.Choice('learning_rate’,
              values=[1e-2, 1e-3, 1e-4])),
       loss='binary_crossentropy’,
       metrics=METRICS)
return model
```

For example the code above demonstrate the way the set up the variables in model structures when initiating the hypermodel object.

HYPERMODEL Object



The three optimisation method

1. **Random search:**
For every dimension in your search space, this algorithm will select a random value, train the model, and report the results.

2. **Bayesian optimization:**
Viewing hyperparameters tuning as the optimization of a black-box function, and using Bayes' rule for optimization.

3. **Hyperband:**
This one attempts to reduce the total tuning time by running experiments very shortly, then only taking the best of them for longer training, in a competition-style fashion.

Search Space

1. **Boolean values:**
Which are set to true or false

2. **Choice values:**
Which represent an array of choices from which one value is chosen

3. **Fixed values:**
Which aren't tunable but rather are fixed as they are.

4. **Float values:**
Which represent floating-point values (such as the learning rate above).

5. **Integer values:**
Which represent integer values to be tuned (such as the number of layers).

And you can also constrain the searching: by setting a maximum number of trials, you can tell the tuner to cut off tuning after some time.

HSBC

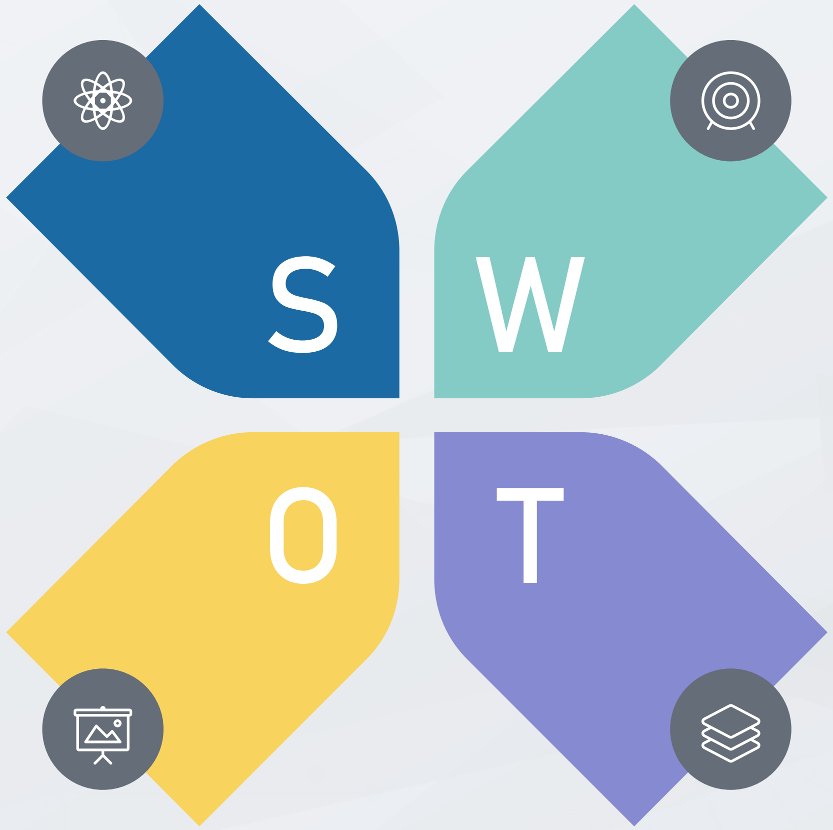
SECTION 4

SWOT & Future Work

Looking at the SWOT analysis of the project and discussing on possible extensions if I have the chance and the time to continue working on this project at HSBC

PART 4: SWOT & Future Work

Strengths, Weaknesses, Opportunities and Threats analysis



Strengths

Integrated toolkits that save time on coding

Automated framework that work as a guidance

Expandable framework that constantly update

Enable knowledge sharing across team

Weaknesses

Currently only covered binary classification problem

Have not covered everything about data cleaning

Not optimised for other teams' project yet

Have not wrapped as a pip-installable package

Opportunities

HSBC expanding investment in data science

Diversified projects require a centralized system

Unifying tools across different labs in DSE

Threats

The algorithm may go out-dated

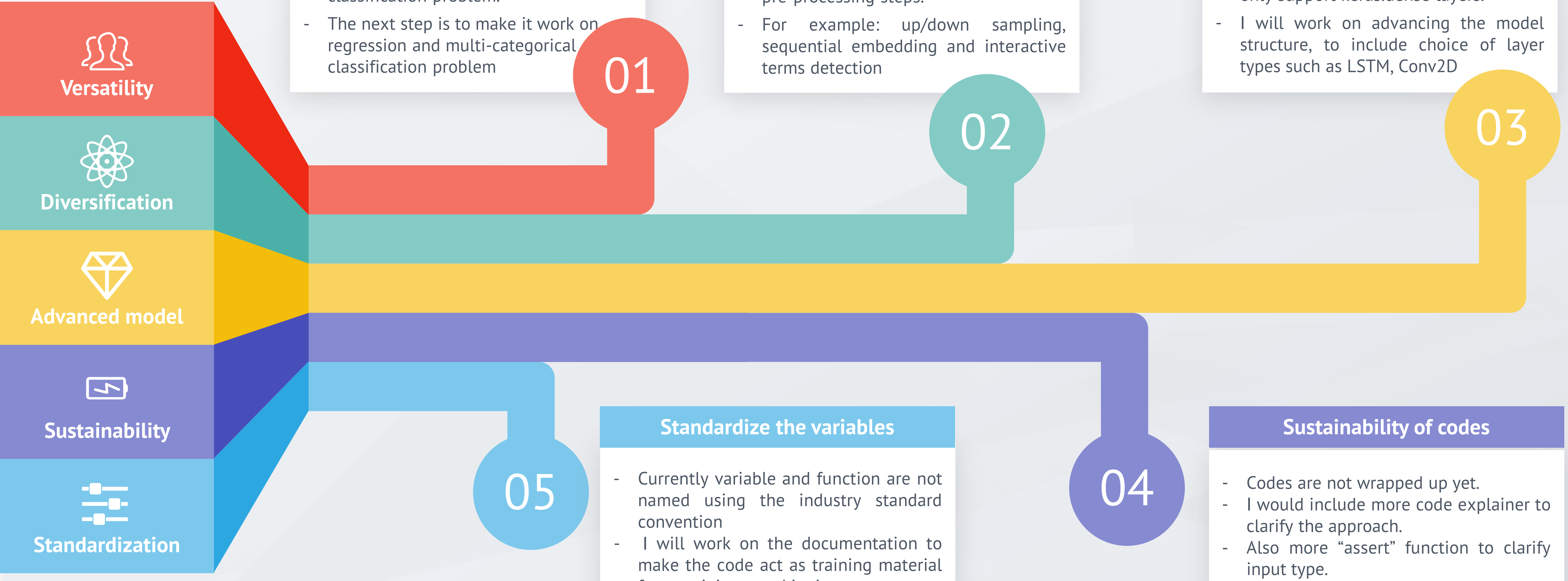
More PyTorch implementation going on

Unifying the whole group may be difficult

PART 4: SWOT & Future Work

What to do next?

Considering the time limits of 5 weeks on project, there are still some many areas of improvements on this auto-ML framework. If I have the chance to work as a full time analyst here, below are the areas I will continue working on



More Versatile Framework

- Currently only works on binary classification problem.
- The next step is to make it work on regression and multi-categorical classification problem

Diversify Functionality

- Expand the methods available in each pre-processing steps.
- For example: up/down sampling, sequential embedding and interactive terms detection

More Advanced Keras structure

- Currently, the hyperparameter tuning only support keras.dense layers.
- I will work on advancing the model structure, to include choice of layer types such as LSTM, Conv2D

Standardize the variables

- Currently variable and function are not named using the industry standard convention
- I will work on the documentation to make the code act as training material for new joiners and juniors.

Sustainability of codes

- Codes are not wrapped up yet.
- I would include more code explainer to clarify the approach.
- Also more "assert" function to clarify input type.

Thank You For Listening

Question & Answer

HSBC

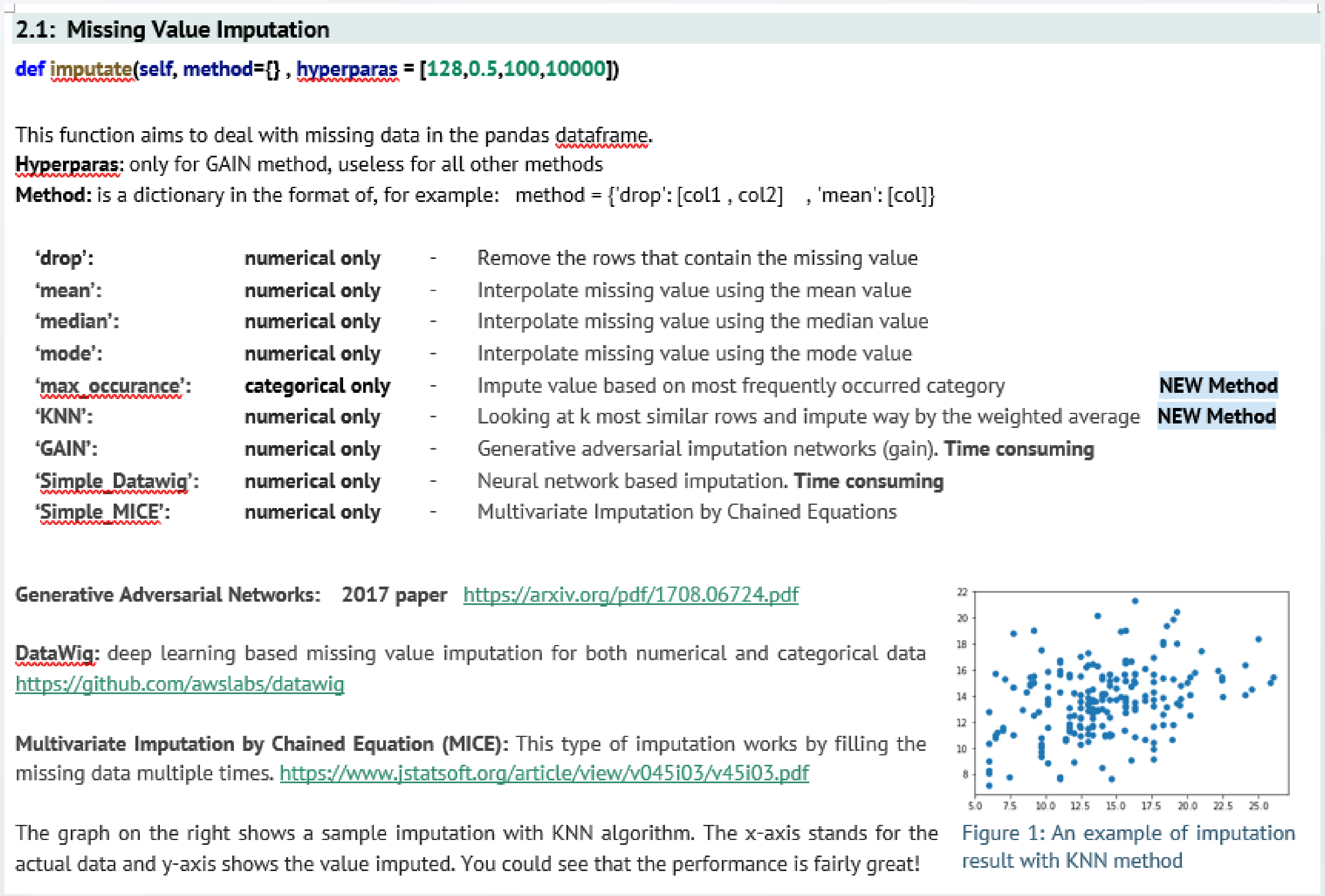
Appendix

Documentation

Detailed documentation on how to use the functions

Data Science and Engineering
HSBC SUMMER INTERNSHIP

Appendix: Codes Documentation



2.1: Missing Value Imputation

```
def imputate(self, method={} , hyperparas = [128,0.5,100,10000])
```

This function aims to deal with missing data in the pandas dataframe.
Hyperparas: only for GAIN method, useless for all other methods
Method: is a dictionary in the format of, for example: method = {'drop': [col1 , col2] , 'mean': [col]}

'drop':	**numerical only**	-	Remove the rows that contain the missing value	
'mean':	**numerical only**	-	Interpolate missing value using the mean value	
'median':	**numerical only**	-	Interpolate missing value using the median value	
'mode':	**numerical only**	-	Interpolate missing value using the mode value	
'max_occurance':	**categorical only**	-	Impute value based on most frequently occurred category	**NEW Method**
'KNN':	**numerical only**	-	Looking at k most similar rows and impute way by the weighted average	**NEW Method**
'GAIN':	**numerical only**	-	Generative adversarial imputation networks (gain). **Time consuming**	
'Simple_Datawig':	**numerical only**	-	Neural network based imputation. **Time consuming**	
'Simple_MICE':	**numerical only**	-	Multivariate Imputation by Chained Equations	

Generative Adversarial Networks: **2017 paper** https://arxiv.org/pdf/1708.06724.pdf

DataWig: deep learning based missing value imputation for both numerical and categorical data https://github.com/awslabs/datawig

Multivariate Imputation by Chained Equation (MICE): This type of imputation works by filling the missing data multiple times. https://www.jstatsoft.org/article/view/v045i03/v45i03.pdf

The graph on the right shows a sample imputation with KNN algorithm. The x-axis stands for the actual data and y-axis shows the value imputed. You could see that the performance is fairly great!

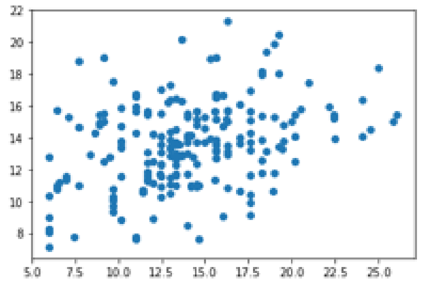
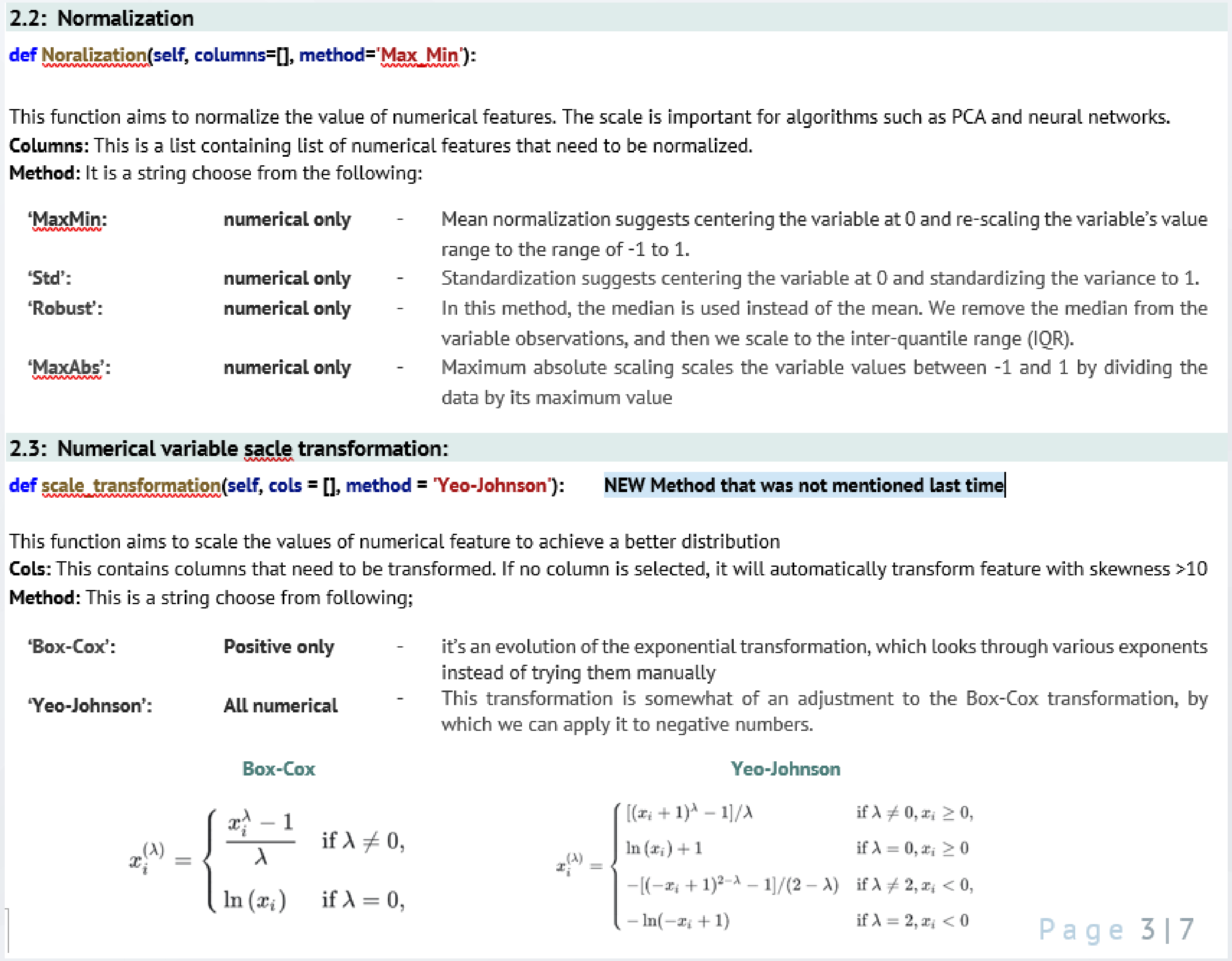


Figure 1: An example of imputation result with KNN method

Appendix: Codes Documentation



2.2: Normalization

```
def Noralization(self, columns=[], method='Max_Min'):
```

This function aims to normalize the value of numerical features. The scale is important for algorithms such as PCA and neural networks.
Columns: This is a list containing list of numerical features that need to be normalized.
Method: It is a string choose from the following:

‘MaxMin:	numerical only	-	Mean normalization suggests centering the variable at 0 and re-scaling the variable’s value range to the range of -1 to 1.
‘Std’:	numerical only	-	Standardization suggests centering the variable at 0 and standardizing the variance to 1.
‘Robust’:	numerical only	-	In this method, the median is used instead of the mean. We remove the median from the variable observations, and then we scale to the inter-quantile range (IQR).
‘MaxAbs’:	numerical only	-	Maximum absolute scaling scales the variable values between -1 and 1 by dividing the data by its maximum value

2.3: Numerical variable sacle transformation:

```
def scale_transformation(self, cols = [], method = 'Yeo-Johnson'):
```
NEW Method that was not mentioned last time

This function aims to scale the values of numerical feature to achieve a better distribution
Cols: This contains columns that need to be transformed. If no column is selected, it will automatically transform feature with skewness >10
Method: This is a string choose from following;

‘Box-Cox’:	Positive only	-	it’s an evolution of the exponential transformation, which looks through various exponents instead of trying them manually
‘Yeo-Johnson’:	All numerical	-	This transformation is somewhat of an adjustment to the Box-Cox transformation, by which we can apply it to negative numbers.

Box-Cox

$$x_i^{(\lambda)} = \begin{cases} \dfrac{x_i^{\lambda} - 1}{\lambda} & \text{if } \lambda \neq 0, \\ \ln(x_i) & \text{if } \lambda = 0, \end{cases}$$

Yeo-Johnson

$$x_i^{(\lambda)} = \begin{cases} [(x_i + 1)^{\lambda} - 1]/\lambda & \text{if } \lambda \neq 0, x_i \geq 0, \\ \ln(x_i) + 1 & \text{if } \lambda = 0, x_i \geq 0 \\ -[(-x_i + 1)^{2-\lambda} - 1]/(2 - \lambda) & \text{if } \lambda \neq 2, x_i < 0, \\ -\ln(-x_i + 1) & \text{if } \lambda = 2, x_i < 0 \end{cases}$$

Page 3 | 7

Appendix: Codes Documentation

2.4: Outlier removal

```
def outlier_removal(self,columns=[],limits=[0.05,0.95],method='Trim'):
```

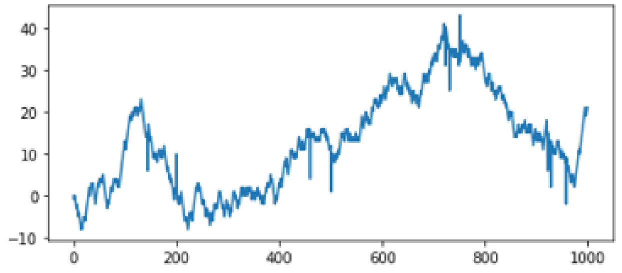
This function aims to remove the outlier with the give percentage in order to achieve a better ML fitting result
columns: This is a list containing list of numerical features that need to check the outliers.
limits: The upper and lower percentage of word you want to remove. **Not applicable** on Gaussian method.
method: It is a string choose from the following:

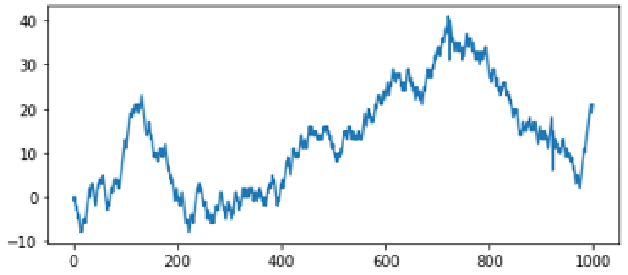
'Trim':	**numerical only**	-	Trimming (or truncation) merely means removing outliers from the dataset; what we need here is just to decide on a metric to determine outliers.
'Inter-quantal':	**numerical only**	-	In this rule, the boundaries are determined using IQR proximity rules. Values bigger or smaller than the chosen value are replaced by this value.
'Gaussian':	**numerical only**	-	This method sets the boundaries with values according to the mean and standard deviation
'Quantiles':	**numerical only**	-	The boundaries are determined using the quantiles, through which you can specify any percentage you want

```
def hampel_filter(self,columns, window_size=10, n_sigmas=3):
```
NEW Method that was not mentioned last time

This function removes outliers **only on time-series object!** The Hampel filter is a member of the class of decsion filters that replaces the central value in the data window with the median if it lies far enough from the median to be deemed an outlier.
Link: https://link.springer.com/article/10.1186/s13634-016-0383-6#:~:text=The%20Hampel%20filter%20is%20a,to%20be%20deemed%20an%20outlier.

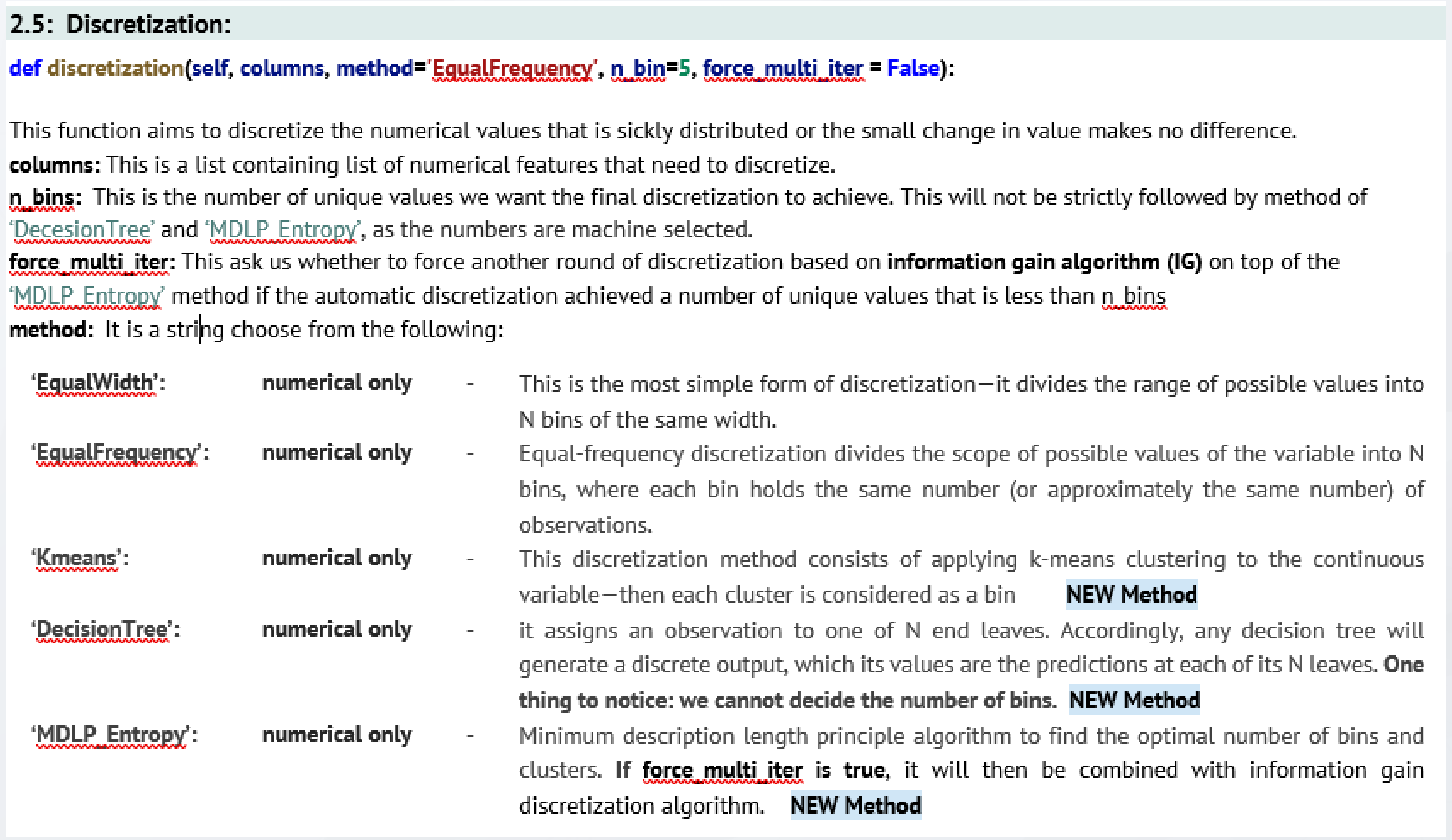
columns: This is a list containing list of numerical features that need to check the outliers.
window_size: This is the number of historical values to look at
n_sigmas: hyperparameter for sensitivity





The figure above show how Hampel filter remove the outliers in the time series.

Appendix: Codes Documentation

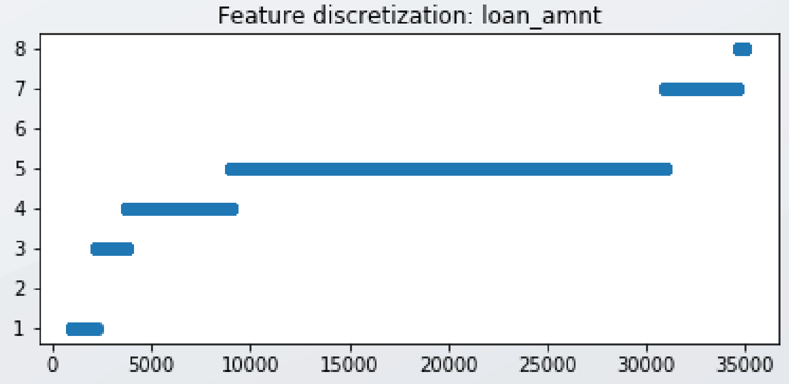


2.5: Discretization:

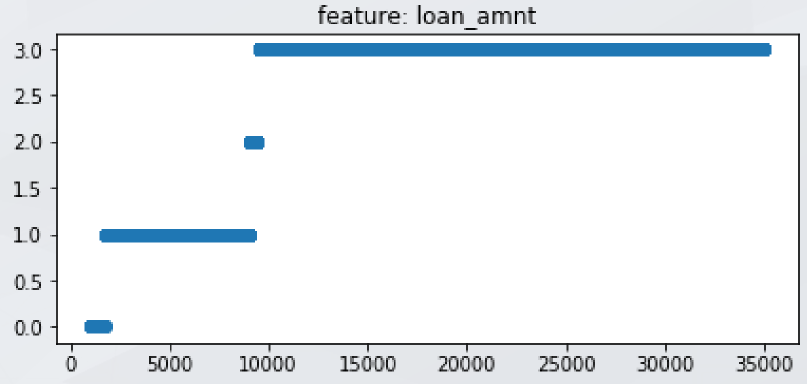
```
def discretization(self, columns, method='EqualFrequency', n_bin=5, force_multi_iter = False):
```

This function aims to discretize the numerical values that is sickly distributed or the small change in value makes no difference.
columns: This is a list containing list of numerical features that need to discretize.
n_bins: This is the number of unique values we want the final discretization to achieve. This will not be strictly followed by method of 'DecesionTree' and 'MDLP_Entropy', as the numbers are machine selected.
force_multi_iter: This ask us whether to force another round of discretization based on **information gain algorithm (IG)** on top of the 'MDLP_Entropy' method if the automatic discretization achieved a number of unique values that is less than n_bins
method: It is a string choose from the following:

'EqualWidth':	**numerical only**	-	This is the most simple form of discretization—it divides the range of possible values into N bins of the same width.
'EqualFrequency':	**numerical only**	-	Equal-frequency discretization divides the scope of possible values of the variable into N bins, where each bin holds the same number (or approximately the same number) of observations.
'Kmeans':	**numerical only**	-	This discretization method consists of applying k-means clustering to the continuous variable—then each cluster is considered as a bin **NEW Method**
'DecisionTree':	**numerical only**	-	it assigns an observation to one of N end leaves. Accordingly, any decision tree will generate a discrete output, which its values are the predictions at each of its N leaves. **One thing to notice: we cannot decide the number of bins.** **NEW Method**
'MDLP_Entropy':	**numerical only**	-	Minimum description length principle algorithm to find the optimal number of bins and clusters. **If force_multi_iter is true,** it will then be combined with information gain discretization algorithm. **NEW Method**



MDLP & IG method



Decision tree method

Appendix: Codes Documentation

2.6: Linear interpolation

```
def linear_interpolation(self, columns, method='fill_between'):
```

This function interpolate missing values **only on time-series object!**
columns: This is a list containing list of numerical features that need to be interpolated.
method: It is a string choose from the following

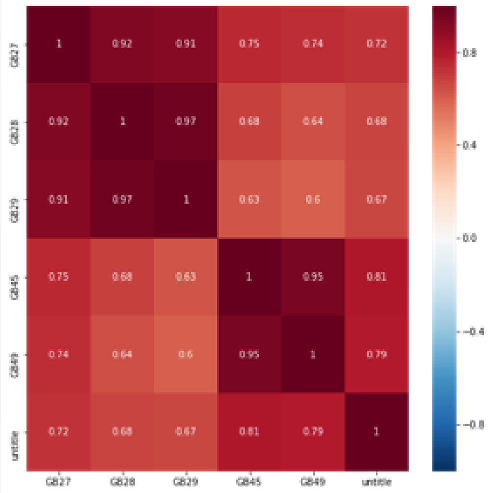
'fill_between':	**numerical only**	-	Fill the missing value based on the upper and lower non-missing data and interpolate the values
'EMW':	**numerical only**	-	Fill the missing value based on the exponentially weighted average of historical 5 days values

2.7: High correlation detection

```
def get_top_abs_correlations(self, n=5,only_numeric=True):
```

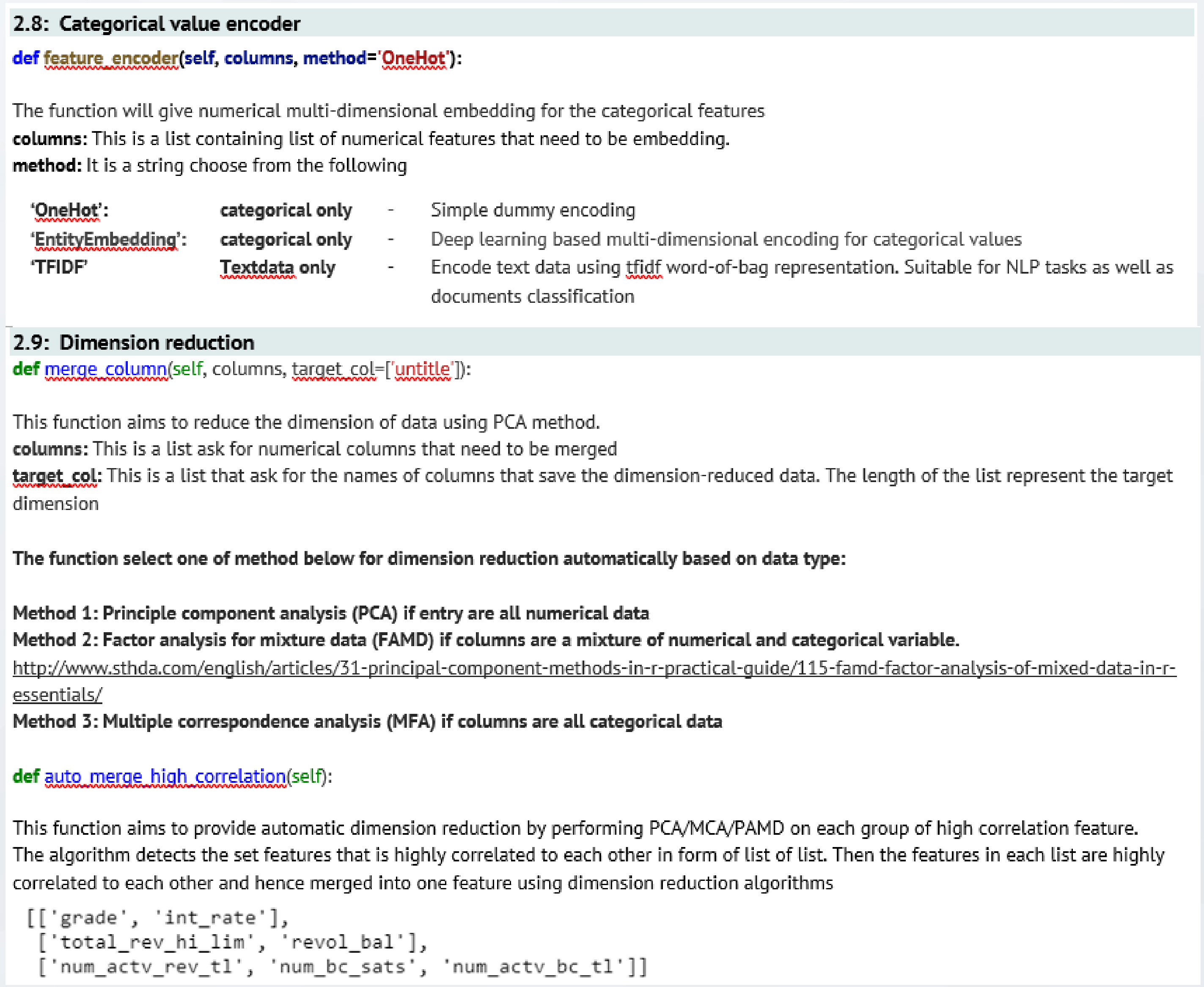
The function will visualize the correlation if the data dimension<=10. It will return pairs of features that has highest correlation

n: Top n highest correlated features are shown
only_numeric: this asks whether we only look numerical columns' correlations



```
GB28  GB29        0.974406
GB45  GB49        0.947984
GB27  GB28        0.921874
      GB29        0.909907
GB45  untitle     0.805477
dtype: float64
```

Appendix: Codes Documentation



2.8: Categorical value encoder

def feature_encoder(self, columns, method='OneHot'):

The function will give numerical multi-dimensional embedding for the categorical features
columns: This is a list containing list of numerical features that need to be embedding.
method: It is a string choose from the following

'OneHot':	categorical only	-	Simple dummy encoding
'EntityEmbedding':	categorical only	-	Deep learning based multi-dimensional encoding for categorical values
'TFIDF'	Textdata only	-	Encode text data using tfidf word-of-bag representation. Suitable for NLP tasks as well as documents classification

2.9: Dimension reduction

def merge_column(self, columns, target_col=['untitle']):

This function aims to reduce the dimension of data using PCA method.
columns: This is a list ask for numerical columns that need to be merged
target_col: This is a list that ask for the names of columns that save the dimension-reduced data. The length of the list represent the target dimension

The function select one of method below for dimension reduction automatically based on data type:

Method 1: Principle component analysis (PCA) if entry are all numerical data
Method 2: Factor analysis for mixture data (FAMD) if columns are a mixture of numerical and categorical variable.
http://www.sthda.com/english/articles/31-principal-component-methods-in-r-practical-guide/115-famd-factor-analysis-of-mixed-data-in-r-essentials/
Method 3: Multiple correspondence analysis (MFA) if columns are all categorical data

def auto_merge_high_correlation(self):

This function aims to provide automatic dimension reduction by performing PCA/MCA/PAMD on each group of high correlation feature. The algorithm detects the set features that is highly correlated to each other in form of list of list. Then the features in each list are highly correlated to each other and hence merged into one feature using dimension reduction algorithms

```
[['grade', 'int_rate'],
 ['total_rev_hi_lim', 'revol_bal'],
 ['num_actv_rev_tl', 'num_bc_sats', 'num_actv_bc_tl']]
```

Appendix: Codes Documentation

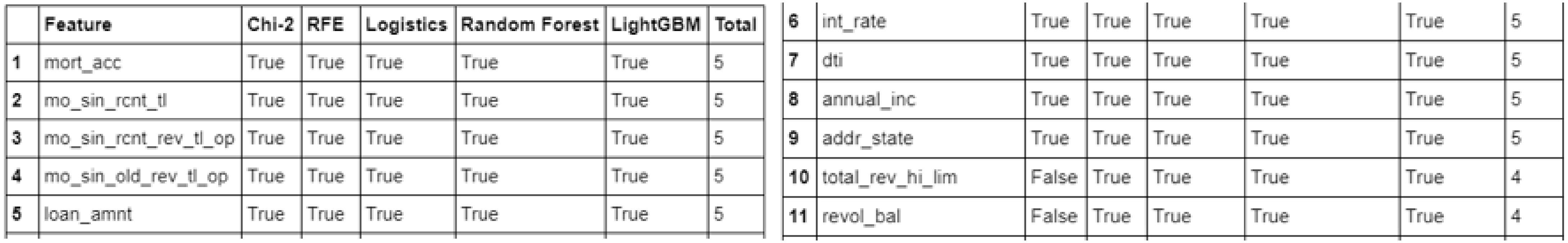
2.10: Auto Feature Selection

```
def auto_feature_selection(self, n_features = 'auto'):
```

This function aims to selected the best N features from all available features using a combination of different algorithms.
n_features: The number of feature we want to keep. If it equals 'auto', half of the features are selected

Feature selection are based on following algorithms simultaneously and then decided by majority vote

- **Chi-2**
- **Recursive feature elimination (RFE)** http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html
- **Embedded LASSO regression** or **L1 penalized Logistic Regression**
- **Random Forest Importance**
- **LightGBM**: Light GBM is a gradient boosting framework that uses tree based learning algorithm. https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc



	Feature	Chi-2	RFE	Logistics	Random Forest	LightGBM	Total
1	mort_acc	True	True	True	True	True	5
2	mo_sin_rcnt_tl	True	True	True	True	True	5
3	mo_sin_rcnt_rev_tl_op	True	True	True	True	True	5
4	mo_sin_old_rev_tl_op	True	True	True	True	True	5
5	loan_amnt	True	True	True	True	True	5
6	int_rate	True	True	True	True	True	5
7	dti	True	True	True	True	True	5
8	annual_inc	True	True	True	True	True	5
9	addr_state	True	True	True	True	True	5
10	total_rev_hi_lim	False	True	True	True	True	4
11	revol_bal	False	True	True	True	True	4