CLO Insight Documentation

Release 1.0.0.0

DBRS

Oct 22, 2023

CONTENTS

PDF

Part I

DBRS Morningstar CLO Insight Predictive Model

CHAPTER

ONE

SOURCE CODE WHITE PAPER

DBRS Morningstar is releasing this white paper as part of its ongoing efforts to provide greater transparency regarding its methodologies and models to market participants. This white paper should be read in conjunction with the related public methodology: Global Methodology for Rating CLOs and Corporate CDOs

CLO Insight is written in Python and utilizes a variety of open source packages to perform its computations:

https://pandas.pydata.org/docs/

https://numpy.org/doc/

https://docs.scipy.org/doc/scipy/

CHAPTER

TWO

NAVIGATING THE SOURCE CODE

Each entry point method is exposed to the user interface via APIs. The details of these are described by the developers in the source code doc strings. This white paper is produced programmatically via these doc strings. The documentation for each API entry method is linked below:

model_clo_insight.app.step_1_parse_pool

model_clo_insight.app.step_2_run_indicative_pool

model_clo_insight.app.step_3_build_trading_scenarios

model_clo_insight.app.step_4a_generate_cf_scenarios

model_clo_insight.app.step_4b_run_trading_scenarios

model_clo_insight.app.step_5_final_results

Descriptions of the internal computational methods can be found in:

model_clo_insight.engines

Descriptions of the statistical methods used to analyse the cumulative default distribution can be found in:

model_clo_insight.analysis

Descriptions of the data structures used to communicate externally with the API methods can be found in:

model_clo_insight.clo_external_data_contracts

Descriptions of the data structures used by the internal computational methods can be found in:

model_clo_insight.clo_internal_objects

CHAPTER

THREE

DISCLAIMER

 The information upon which DBRS Morningstar credit ratings and other types of credit opinions and reports are based is obtained by DBRS Morningstar from sources DBRS Morningstar believes to be reliable. DBRS Morningstar does not audit the information it receives in connection with the analytical process, and it does not and cannot independently verify that information in every instance. The extent of any factual investigation or independent verification depends on facts and circumstances. DBRS Morningstar credit ratings, other types of credit opinions, reports and any other information provided by DBRS Morningstar are provided "as is" and without representation or warranty of any kind and DBRS Morningstar assumes no obligation to update any such ratings, opinions, reports or other information. DBRS Morningstar hereby disclaims any representation or warranty, express or implied, as to the accuracy, timeliness, completeness, merchantability, fitness for any particular purpose or non-infringement of any of such information. In no event shall DBRS Morningstar or its directors, officers, employees, independent contractors, agents, affiliates and representatives (collectively, DBRS Morningstar Representatives) be liable (1) for any inaccuracy, delay, loss of data, interruption in service, error or omission or for any damages resulting therefrom, or (2) for any direct, indirect, incidental, special, compensatory or consequential damages arising from any use of credit ratings, other types of credit opinions and reports or arising from any error (negligent or otherwise) or other circumstance or contingency within or outside the control of DBRS Morningstar or any DBRS Morningstar Representative, in connection with or related to obtaining, collecting, compiling, analyzing, interpreting, communicating, publishing or delivering any such information. IN ANY EVENT, TO THE EXTENT PERMITTED BY LAW, THE AGGREGATE LIABILITY OF DBRS MORNINGSTAR AND THE DBRS MORNINGSTAR REPRESENTATIVES FOR ANY REASON WHATSOEVER SHALL NOT EXCEED THE GREATER OF (A) THE TOTAL AMOUNT PAID BY THE USER FOR SERVICES PROVIDED BY DBRS MORNINGSTAR DURING THE TWELVE (12) MONTHS IMMEDIATELY PRECEDING THE EVENT GIVING RISE TO LIABILITY, AND (B) U.S. $100. DBRS Morningstar does not act as a fiduciary or an investment advisor. DBRS Morningstar does not provide investment, financial or other advice. Credit ratings, other types of credit opinions and other analysis and research issued by DBRS Morningstar (a) are, and must be construed solely as, statements of opinion and not statements of fact as to credit worthiness, investment, financial or other advice or recommendations to purchase, sell or hold any securities; (b) do not take into account your personal objectives, financial situations or needs; (c) should be weighed, if at all, solely as one factor in any investment or credit decision; (d) are not intended for use by retail investors; and (e) address only credit risk and do not address other investment risks, such as liquidity risk or market volatility risk. Accordingly, credit ratings, other types of credit opinions and other analysis and research issued by DBRS Morningstar are not a substitute for due care and the study and evaluation of each investment decision, security or credit that one may consider making, purchasing, holding, selling, or providing, as applicable. A report with respect to a DBRS Morningstar credit rating or other credit opinion is neither a prospectus nor a substitute for the information assembled, verified and presented to investors by the issuer and its agents in connection with the sale of the securities. DBRS Morningstar may receive compensation for its credit ratings and other credit opinions from, among others, issuers, insurers, guarantors and/or underwriters of debt securities. This publication may not be reproduced, retransmitted or distributed in any form without the prior written consent of DBRS Morningstar. ALL DBRS MORNINGSTAR CREDIT RATINGS AND OTHER TYPES OF CREDIT OPINIONS ARE SUBJECT TO DEFINITIONS, LIMITATIONS, POLICIES AND METHODOLOGIES THAT ARE AVAILABLE ON https://www.dbrsmorningstar.com. Users may, through hypertext or other computer links, gain access to or from websites operated by persons other than DBRS Morningstar. Such hyperlinks or other computer links are provided for convenience only. DBRS Morningstar does not endorse the content, the operator or operations of third party websites. DBRS Morningstar is not responsible for the content or operation of such third

party websites and DBRS Morningstar shall have no liability to you or any other person or entity for the use of third party websites.

Please Note

Credit rating services offered by DBRS Morningstar are separate, independent, and distinct from the products and services offered by Morningstar, Inc. and its non-credit rating subsidiaries. DBRS Morningstar credit ratings are formed and disseminated based on established methodologies, models and criteria Methodology(ies) that apply to entities and securities that we rate, including corporate finance issuers, financial institutions, insurance companies, public finance and sovereign entities as well as Structured Finance transactions. From time to time, DBRS Morningstar may permit Morningstar, Inc. and its non-credit rating subsidiaries to use, modify, display and allow for interactive use DBRS Morningstar Analytics designed, created and published by DBRS Morningstar.

Part II

clo-insight

CHAPTER

FOUR

MODEL_CLO_INSIGHT PACKAGE

The core library of the model which contains the model logic and all related quantitative aspects.

An easy way to navigate the code is to start at the documentation for each of the `API Endpoint` methods in *model_clo_insight.app*

Within the documentation for those `API Endpoint` methods, there are links to the actual `Computational Engine` methods contained in *model_clo_insight.engines*

4.1 Subpackages

4.1.1 model_clo_insight.analysis package

Analytics Package

Used for quantitative and statistical forumlas used in the rating process. In this model that would be the statistical analysis of the default / loss distributions, and pool metrics related to hypo pool generation.

4.1.1.1 Submodules

model_clo_insight.analysis.data module

Module for methods related to dataframe aggregation

stratify(*df*, *group_by_col*, *count_col*, *par_col*, *is_sort=True*, *top_n=None*)
Computes stratification of a specified dataframe column

Parameters

- **df** (*pd.DataFrame*) – dataframe column is taken from
- **group_by_col** (*str*) – column to use for the groupby operation
- **count_col** (*str*) – column to use for counting
- **par_col** (*str*) – column to use for par
- **is_sort** (*bool*) – optional; if true, sort the results column by percent (high to low)
- **top_n** (*int*) – optional; limit the results to the top n items

Returns Count, Par, Percent as columns and rows are categories

Return type df (pd.DataFrame)

w_avg(*df*, *values*, *weights*)
Computes weighted average of a specified dataframe column

Parameters

- **df** (*pd.DataFrame*) – dataframe column is taken from
- **values** (*str*) – column name for values
- **weights** (*str*) – column name for weights

Returns computed weighted average

Return type float

model_clo_insight.analysis.pool_metrics module

Module that contains methods that compute various pool metrics. These are used extensively in the generation of stressed modeling pools (hypo pools)

calc_adj_class_bdr(*class_bdr*, *rvst_target_par*, *collat_prin_amt*, *warr*)
Computes Adjusted Class Break-even Default Rate

- Used for CDO Monitor Test
- Adjusts Class Break-even Default Rate for par build / erosion

Parameters

- **class_bdr** (*float*) – Result of calc_class_bdr()
- **rvst_target_par** (*float*) – Reinvestment target par as per the transaction documents
- **coll_prin_amt** (*float*) – Aggregate principal amount of the current portfolio
- **warr** (*float*) – Weighted average recovery rate for the current portfolio

Returns Class Adjusted BDR

Return type float

calc_asset_side(*run_mode*, *pool*, *wal*, *coefs*, *WARF_sp*)
Wrapper for the underlying methods that computes all the pool metrics for the CDO Monitor SDR

Parameters

- **run_mode** (*str*) – routing for EPDR vs SP_WARF
- **pool** (*list[clo_internal_objects.Obligor]*) – pool of obligors to calculate the metric on
- **wal** (*float*) – Weighted average life of the current portfolio
- **coefs** (*list[float]*) – coefficients for the CDO Monitor SDR formula
- **WARF_sp** (*dict*) – lookup table for weighted average rating factor

Returns Class SDR, diagnostics dictionary

Return type tuple

calc_asset_side_A(*pool*, *wal*, *coefs*)
Calls each pool metric required for the CDO Monitor SDR (for IDT PD approach)

Parameters

- **pool** (*list[clo_internal_objects.Obligor]*) – pool of obligors to calculate the metric on
- **wal** (*float*) – Weighted average life of the current portfolio
- **coefs** (*list[float]*) – coefficients for the CDO Monitor SDR formula

Returns Class SDR, diagnostics dictionary

Return type tuple

calc_asset_side_B(*pool*, *wal*, *coefs*, *WARF_sp*)

Calls each pool metric required for the CDO Monitor SDR (for WARF approach)

Parameters

- **pool** (*list[clo_internal_objects.Obligor]*) – pool of obligors to calculate the metric on
- **wal** (*float*) – Weighted average life of the current portfolio
- **coefs** (*list[float]*) – coefficients for the CDO Monitor SDR formula
- **WARF_sp** (*dict*) – lookup table for weighted average rating factor

Returns Class SDR, diagnostics dictionary

Return type tuple

calc_class_bdr(*was*, *warr*, *coef*)

Computes Class Break-even Default Rate

- Used for CDO Monitor Test
- Designed to proxy a full BDR analysis of the highest priority class in the capital structure

Parameters

- **was** (*float*) – Weighted average spread for the current portfolio
- **warr** (*float*) – Weighted average recovery rate for the current portfolio
- **coef** (*list[float]*) – coefficients for the CDO Monitor BDR formula (transaction specific)

Returns Class BDR

Return type float

calc_class_sdr_exp_def_rate(*exp_def_rate*, *def_rate_dispersion*, *ob_div_measure*, *ind_div_measure*, *reg_div_measure*, *wal*, *coefs*)

Computes Class Scenario Default Rate: Method A (Expected Default Rate)

- Used for CDO Monitor Test
- Proxy for cumulative default hurdle rates from CDO Evaluator
- Relies on the results of the following functions:
  - calc_exp_def_rate
  - calc_def_rate_dispersion
  - calc_ind_div_measure
  - calc_ind_div_measure

Parameters

- **exp_def_rate** (*float*) – Expected default rate of the current portfolio
- **def_rate_dispersion** (*float*) – Default rate dispersion of the current portfolio
- **ob_div_measure** (*float*) – Obligor diversity measure of the current portfolio
- **ind_div_measure** (*float*) – Industry diversity measure of the current portfolio
- **reg_div_measure** (*float*) – Regional diversity measure of the current portfolio
- **wal** (*float*) – Weighted average life of the current portfolio
- **coefs** (*list[float]*) – coefficients for the CDO Monitor SDR formula

Returns Class SDR

Return type float

calc_class_sdr_sp_warf(*exp_def_rate*, *def_rate_dispersion*, *ob_div_measure*, *ind_div_measure*, *reg_div_measure*, *wal*, *coefs*)

Computes Class Scenario Default Rate: Method B (Using WARF)

- Used for CDO Monitor Test
- Proxy for cumulative default hurdle rates from CDO Evaluator
- Relies on the results of the following functions:
  - calc_exp_sp_warf
  - calc_def_rate_dispersion
  - calc_ind_div_measure
  - calc_ind_div_measure

Parameters

- **exp_def_rate** (*float*) – Expected default rate of the current portfolio
- **def_rate_dispersion** (*float*) – Default rate dispersion of the current portfolio
- **ob_div_measure** (*float*) – Obligor diversity measure of the current portfolio
- **ind_div_measure** (*float*) – Industry diversity measure of the current portfolio
- **reg_div_measure** (*float*) – Regional diversity measure of the current portfolio
- **wal** (*float*) – Weighted average life of the current portfolio
- **coefs** (*list[float]*) – coefficients for the CDO Monitor SDR formula

Returns Class SDR

Return type float

calc_credit_dispersion(*obligors*, *risk_score_table*)

calculates the rating disperison of a pool from a list of obligor objects.

Parameters

- **obligors** (*dict[dataclasses]*) – list of obligors
- **risk_score_table** (*dict*) – rating score table

Returns rating dispersion of the pool

Return type float

calc_def_rate_dispersion(*pool*, *exp_def_rate*)
Computes Absolute Deviation of PD for a Pool via IDT

- Used in Class Scenario Default Rate calcs, method A

Parameters

- **pool** (*list[clo_internal_objects.Obligor]*) – pool of obligors to calculate the metric on
- **exp_def_rate** (*float*) – Expected pool default rate (EPDR) via calc_exp_def_rate()

Returns Default rate dispersion (DRD)

Return type float

calc_dscore(*obligors*, *tables*)
Computes the Diversity Score of a pool of obligors based on the DBRSM Industry classification

Parameters

- **obligors** (*list[clo_internal_objects.Obligor]*) – pool of obligors to calculate the metric on
- **tables** (*dict*) – dictionary of various lookup tables used in the calculation

Returns Diversity Score of the pool

Return type float

calc_exp_def_rate(*pool*)
Computes E(default rate) via IDT for a Pool

- Used in Class Scenario Default Rate calcs
- IDT lookup uses rating and tenor

Parameters **pool** (*list[clo_internal_objects.Obligor]*) – pool of obligors to calculate the metric on

Returns Expected pool default rate (EPDR)

Return type float

calc_exp_sp_warf(*pool*, *WARF_sp*)
Computes E(default rate) for a Pool via WARF

- Used in Class Scenario Default Rate calcs for certain middle market transactions
- WARF does not have a tenor component

Parameters

- **pool** (*list[clo_internal_objects.Obligor]*) – pool of obligors to calculate the metric on
- **WARF_sp** (*dict*) – lookup table for weighted average rating factor

Returns Expected pool default rate (EPDR)

Return type float

calc_ind_div_measure(*pool*)
Computes Industry Diversity Metric

- Used in the CDO Monitor calculations
- Based on the Herfindahl index

Parameters **pool** (*list[clo_internal_objects.Obligor]*) – pool of obligors to calculate the metric on

Returns Industry diversity metric (IDM)

Return type float

calc_ob_div_measure(*pool*)
Computes Obligor Diversity Metric

- Used in the CDO Monitor calculations
- Based on the Herfindahl index

Parameters **pool** (*list[clo_internal_objects.Obligor]*) – pool of obligors to calculate the metric on

Returns Obligor diversity metric (ODM)

Return type float

calc_riskscore(*obligors*, *tables*)
Computes the weighted average DBRSM Risk Score of a pool of obligors based on a string DBRSM rating

Parameters

- **obligors** (*list[clo_internal_objects.Obligor]*) – pool of obligors to calculate the metric on
- **tables** (*dict*) – dictionary of various lookup tables used in the calculation

Returns par weighted average DBRSM Risk Score of the pool

Return type float

calc_riskscore_numerical_rtg(*obligors*, *risk_score_table*)
Computes the WA DBRSM Risk Score of a pool of obliors based on an integer DBRSM rating

Parameters

- **obligors** (*list[clo_internal_objects.Obligor]*) – pool of obligors to calculate the metric on
- **tables** (*dict*) – dictionary of various lookup tables used in the calculation

Returns par weighted average DBRSM Risk Score of the pool

Return type float

calc_sp_warf_dispersion(*pool*, *exp_sp_warf*, *WARF_sp*)
Computes Absolute Deviation of WA Rating Factor for a Pool via WARF

- Used in Class Scenario Default Rate calcs, method B

Parameters

- **pool** (*list[clo_internal_objects.Obligor]*) – pool of obligors to calculate the metric on

- **exp_sp_warf** (*float*) – Expected pool default rate (EPDR) via calc_exp_sp_warf()
- **WARF_sp** (*dict*) – lookup table for weighted average rating factor

Returns Default rate dispersion (DRD)

Return type float

model_clo_insight.analysis.stats module

Statistical Module for Monte Carlo Default / Loss Distributions

Computes the rating-based cumulative default/loss hurdles, and diagnostic statistics at both the pool and loan level.

_solve_for_rho(*a*, *b*)
Solves for Vasicek correlation for a given mean and variance

Solves for rho within a Vasicek distribution such that the observed mean and variance are at the target

Parameters

- **a** (*float*) – mean
- **b** (*float*) – variance

Returns rho single correlation parameter for a Vasicek distribution

Return type float

_variance_func(*x*, *y*, *z*)
Used in correlation solver

Returns variance for a Vasicek distribution

Return type float

Notes

- Uses Numpy multivariate normal distribution

calc_default_diagnostics(*def_pct_matrix*)
Computes the expected (exposure at default) per loan based on the simulation results

Parameters **def_pct_matrix** (*ndarray*) – simulation results from model_clo_insight.engines.LGD.calc_def_pct_from_amort()

Returns exposure at default per loan

Return type ndarray

calc_rating_percentile(*target_pd*, *loss_dist*)
Computes Rating-Based Cumulative Default / Loss Thresholds

Parameters

- **target_pd** (*float*) – target PD, based on idealized default table (IDT)
- **loss_dist** (*ndarray*) – histogram computed from run_stats()

Returns cumulative default / loss level at or around 1 - target_pd

Return type float

Notes

- Performs an interpolation as pool default/loss distributions can be lumpy
- This algorithm was taken from the CLO Asset Model C# code: CalcStats() line 1319

interpolate_pd_func()
Creates an interpolation mapping function for the DBRSM IDT

Returns pointer to the interpolation mapping function created in this method

Return type function pointer

run_stats(*monte_carlo_loss*, *securities*, *transaction*, *settings*, *methodology_params*)
Computes all required statistical values for the simulation results. Last large calculation in the model

Parameters

- **monte_carlo_loss** (*ndarray*) – raw simulation results
- **securities** (*list[clo_internal_objects.Corp_Loan]*) – List of the loans in the pool
- **transaction** (clo_internal_objects.TransactionData) – Data at the transaction (e.g. securitization) level:
- **settings** (clo_internal_objects.ModelSettings) – settings object, e.g. monte carlo seed, number of trials
- **methodology_params** – (clo_internal_objects.MethodologyParamsDefaults): methodology params w.r.t. the default simulation

Returns results dict, histograms, cumulative default/loss hurdles, moments, diagnostics

Return type dict

Note that an extra element is needed to ensure 100% defaults go into the 100% (not 99%) bucket Also note that we need to convert both the bins & data to float32 to avoid floating point errors when cumulative loss is exact size of a bin (important for lumpy pools)

4.1.2 model_clo_insight.engines package

Computational Engines

This package contains the modules which perform the core analytical computations for CLO Insight

4.1.2.1 Submodules

model_clo_insight.engines.defaults module

Default Pooling Analytics

run_defaults(*obligors*, *loans*, *transaction_data*, *settings*, *methodology_params*)
CLO Insight Pool Default Monte Simulation

This is the `Monte Carlo Simulation` engine for the `Pool Default` analysis. The inputs to this method are all custom classes, and must be constructed earlier in pre-processing. The simulation is a multi-factor model, where each obligor's random shock is a linear combination of:

- `Global Shock`
- `Industry shock`

- `Region Shock`
- `Idiosyncratic Shock`

The coefficients are computed from the asset correlation assumptions within meth_params. In practice, the correlation assumptions for inter- and intra- region are the same, which results in a zero for the coefficient of regional shock. To optimize this engine for speed, all random numbers are drawn inside Numpy earlier, and matrix arithmetic is used to perform the computations.

Algorithm Steps

- (1) Pre-compute factor coefficients from correlation parameters $a + b$
- (2) Setup random number generator
- (3) Simulate n_{trials} independent random normal `Global Shocks` $n_{trials} = a + b$
- (4) Compute lookup arrays
- (5) Simulate n_{trials} x $n_{obligors}$ independent random normal `Idiosyncratic Shocks` $n_{trials} = a * b$
- (6) Simulate n_{trials} x $n_{industries}$ independent random normal `Industry Shocks`
- (7) Calculate n_{trials} x $n_{obligors}$ correlated random normal variables based on linear combination of the `Shocks`
- (8) Calculate default timing per obligor per trial by looking up the simulated shock variable to the obligor's normalized PD curve

Parameters

- **obligors** (clo_internal_objects.Obligor) – pool of obligors
- **loans** (clo_internal_objects.CorpLoan) – list of the loans in the pool
- **transaction_data** (clo_internal_objects.TransactionData) – Data at the `Transaction` (e.g. securitization) level
- **settings** (clo_internal_objects.ModelSettings) – `Settings` object, e.g. Monte Carlo seed, number of trials
- **methodology_params** (clo_internal_objects.MethodologyParamsDefaults) – `Methodology Parameters`

Returns ttd_matrix (n_{trials} x $n_{obligors}$ matrix of simulated default times) and global_shocks (n_{trials} `Global Shock` values)

Return type tuple of numpy.ndarray

Notes

- Dictionaries cannot be used inside of Numpy's vectorized code, so positional lookups need to be determined prior to the simulation

model_clo_insight.engines.hypo_pool module

This Module Contains the Primary Methods for Constructing Stressed Modeling Pools (Hypo Pools)

class HypoResultMonitor(*is_solve*, *riskscore_solution*, *riskscore_realized*, *credit_dispersion*, *adj_bdr*, *bdr*, *sdr*, *sdr_diag*, *obligors*)

Bases: object

Object which stores the result fields for the hypo_pool_formula_constraints() method Descriptions can be found below in *model_clo_insight.engines.hypo_pool.hypo_pool_formula_constraints*

adj_bdr: float

bdr: float

credit_dispersion: float

is_solve: bool

obligors: list

riskscore_realized: float

riskscore_solution: float

sdr: float

sdr_diag: float

hypo_pool_discrete_constraints(*pool_par*, *conc_limit_basis*, *dscore_min*, *ob_max*, *ind_max*, *riskscore_max*, *ob_exceptions*, *ind_exceptions*, *ob_floor*, *ind_floor*, *risk_score_tbl*, *tables*, *force_high=None*, *force_low=None*, *is_simple_industry=False*, *n_simple_industry=40*)

Main entry point for constructing the hypothetical modeling pool where the CQT constraints are an explicit Dscore and risk score (CQT matrix)

General Algorithm

Solver Goal => Pool actual Dscore ~ Dscore constraint (slightly higher)

2 stage solver: - The outer layer (this method) iterates over the concentration limitation exceptions - The inner layer (the solvers.XXX methods) use numerical optimization on the base concentration limitations

Note: the solver performs the rating overlay once the pool percentages are computed

General steps of the algorithm:

1) Calculate scaling factor for cases where the conc limit basis is different from actual pool par, and apply
2) Run solver and check if solution is found (the solvers spin down the base ob/ind concentration limitation, then rating overlay)
3) If no solution is found, iteratively remove the last (smallest) obligor conc limit exception
   a) If Dscore in compliance, solve for partial last (smallest) obligor percentage s.t. Dscore_act approx Dscore_trigger
   b) If Dscore not in compliance, go to next obligor exception
4) If no solution is found and all obligor conc limits are removed, iteratively remove the last (smallest) industry conc limit exception until a solution is found

Parameters

- **pool_par** (*float*) – Aggregate par balance of the current portfolio

- **conc_limit_basis** (*float*) – Basis for concentration limitations; typically defined the same as rvst target par
- **dscore_min** (*float*) – maximum diversity score limit for the pool
- **ob_max** (*float*) – base max obligor concentration limitation
- **ind_max** (*float*) – base max industry concentration limitation
- **riskscore_max** (*float*) – max DBRSM risk score of the pool
- **ob_exceptions** (*list[clo_internal_objects.ConcLimitException]*) – obligor concentration limitations
- **ind_exceptions** (*list[clo_internal_objects.ConcLimitException]*) – industry concentration limitations
- **ob_floor** (*float*) – lowest level that the base obligor concentration limitation is allowed to spin down to
- **ind_floor** (*float*) – lowest level that the base industry concentration limitation is allowed to spin down to
- **risk_score_tbl** (*dict[int, float]*) – lookup table for DBRSM Risk Score
- **tables** (*dict[str, np.ndarray]*) – Lookup tables for Dscore and industry codes
- **force_high** (*tuple*) – high rating (rating integer, percentage) to lock prior to solving for belly ratings
- **force_low** (*tuple*) – low rating (rating integer, percentage) to lock prior to solving for belly ratings
- **is_simple_industry** (*bool*) – if true, industry allocation is done on a repeating sequence. Hypo only iterates on obligor in this case.
- **n_simple_industry** (*int*) – if is_simple_industry is True, this is the number of industries to repeat in sequence

Returns results dictionary

Return type dict

Results Dictionary Keys:

- is_solve: True if solver found solution, False otherwise
- dscore: diversity score calculated on the actual pool
- riskscore: DBRSM risk score calculated on the actual pool
- ob_base: base obligor concentration limitation that the solver found
- ind_base: base industry concentration limitation that the solver found
- ob_exceptions: obligor concentration limit exceptions actually used
- ind_exceptions: industry concentration limit exceptions actually used
- obligors: OrderedDict[objects.Obligor] pool of Obligors.

hypo_pool_formula_constraints(*hypo_pool_diversity*, *run_mode*, *rs_abs_max*, *coef_bdr*, *coef_sdr*, *risk_score_tbl*, *IDT_sp*, *WARF_sp*, *rvst_target_par*, *collat_prin_amt*, *was*, *warr*, *wal*, *rs_bound_lower*, *rs_bound_higher*, *monitor_cushion*, *force_high=None*, *force_low=None*)

Main entry point for constructing the hypothetical modeling pool where a continuous formula is the constraint

General Algorithm

Solver Goal => Solve for the worst rating mixture such that the Monitor BDR ~ Monitor SDR

Notes: - Diversity should be solved for prior to running this method; this method solves for rating mixture

General steps of the algorithm:

1) Check edge case to see if using the max allowable risk score causes BDR < SDR
2) Check edge case to see if using the min allowable risk score causes BDR > SDR
3) Run a bisection optimizer to solver for a rating mixture (via a risk score) such that BDR ~ SDR
4) Iterate the resulting risk score locally in small chunks until BDR > SDR

Parameters

- **hypo_pool_diversity** (*list[clo_internal_objects.Obligor]*) – pool where obligor & industry percentages have been previously solved for a given Dscore
- **run_mode** (*str*) – Expected portfolio default rate or WA Rating Factor
- **rs_abs_max** (*float*) – absolute max risk score as per deal (monitor solver won't create pools any worse than this)
- **coef_bdr** (*list*) – Break even default rate (BDR) coefficients for the transaction
- **coef_sdr** (*list*) – Scenario even default rate (SDR) coefficients for the transaction
- **risk_score_tbl** (*ndarray*) – DBRSM risk score 1-D table
- **IDT_sp** (*ndarray*) – Idealized default 2-D table; sourced from transaction documents
- **WARF_sp** (*ndarray*) – WARF-equivalent 1-D table; sourced from transaction documents
- **was** (*float*) – Target porfolio weighted average spread
- **warr** (*float*) – Target portfolio weighted average recovery
- **wal** (*float*) – Target portfolio weighted average life
- **rs_bound_lower** (*float*) – max risk score boundary (low), required for optimizer
- **rs_bound_higher** (*float*) – max risk score boundary (high), required for optimizer
- **monitor_cushion** (*float*) – optional; can be used to keep a constant cushion between BDR and SDR when solving
- **force_high** (*tuple*) – (rating_number, pct) for higher rtg prior to solving for risk score, force this on pool
- **force_low** (*tuple*) – (rating_number, pct) for lower rtg prior to solving for risk score, force this on pool

Returns instance results data class object

Return type *HypoResultMonitor*

model_clo_insight.engines.pool_builders module

Module for the methods which build hypo pools

build_pool(*ob_max*, *ind_max*, *riskscore_max*, *ob_ex*, *ind_ex*, *risk_score_tbl*, *tables*, *force_high*, *force_low*, *is_simple_industry*, *n_simple_industry*)

Creates a hypothetical pool

General Algorithm

1) Number of obligors is determined based on concentration limitations. Start with exceptions, then use base
2) Rating mixture of the pool is determined based on max DBRSM Risk Score. First, impose credit dispersion via the force_low and force_high parameters. Then solve for rating mixture on the remaining pool such that the total pool risk score is approximately the target riskscore_max
3) Apply industry exceptions starting with largest obligors
4) Apply industries to the remaining obligors using the base max industry limit

Parameters

- **ob_max** (*float*) – base max obligor concentration limitation
- **ind_max** (*float*) – base max industry concentration limitation
- **riskscore_max** (*float*) – max DBRSM risk score of the pool
- **ob_ex** (*list[objects.ConcLimitException]*) – obligor concentration limitations
- **ind_ex** (*list[objects.ConcLimitException]*) – industry concentration limitations
- **risk_score_tb** (*dict[int, float]*) – lookup table for DBRSM Risk Score
- **tables** (*dict*) – lookup tables. Must contain keys: ind_codes, dscore, dbrs_rtg
- **force_high** (*tuple*) – (rating_number, pct) for higher rtg prior to solving for risk score, force this on pool
- **force_low** (*tuple*) – (rating_number, pct) for lower rtg prior to solving for risk score, force this on pool
- **is_simple_industry** (*bool*) – if true, industry allocation is done on a repeating sequence. Hypo only iterates on obligors in this case.
- **n_simple_industry** (*int*) – if is_simple_industry is True, this is the number of industries to repeat in sequence

Returns pool of Obligors. Assumes one dummy security per obligor, these are setup later.

Return type OrderedDict[objects.Obligor]

Notes

This method does not constrain on pool-wide diversity metrics

build_pool_ind_mult_of_ob(*ob_max*, *ind_mult*, *ind_floor*, *riskscore_max*, *ob_ex*, *ind_ex*, *risk_score_tbl*, *tables*, *force_high*, *force_low*, *is_simple_industry*, *n_simple_industry*)

Used When Building a Pool Where Base Industry is a Function of base ob and ind floor

Parameters

- **ob_max** (*float*) – base max obligor concentration limitation

- **riskscore_max** (*float*) – maximum DBRSM risk score constraint
- **ob_ex** (*list[clo_internal_objects.ConcLimitException]*) – obligor concentration limitations
- **ind_ex** (*list[clo_internal_objects.ConcLimitException]*) – industry concentration limitations
- **tables** (*dict[str, np.ndarray]*) – Lookup tables for diversity score and industry codes
- **ind_mult** (*float*) – base industry concentration is a function of this multiple * obligor base conc limit
- **ind_floor** (*float*) – lowest level that the base industry concentration limitation is allowed to spin down to

Returns pool of Obligors. Assumes one dummy security per obligor

Return type OrderedDict[objects.Obligor]

build_rating_mixture(*riskscore_max*, *hypo_pool*, *risk_score_tbl*, *force_high*, *force_low*)

Parameters

- **riskscore_max** (*float*) – maximum DBRSM risk score constraint
- **hypo_pool** (*list[clo_internal_objects.Obligor]*) – pool of obligors to calculate the metric on
- **risk_score_tbl** (*dict[int, float]*) – lookup table for DBRSM Risk Score
- **force_high** (*tuple*) – (rating_number, pct) for higher rtg prior to solving for risk score, force this on pool
- **force_low** (*tuple*) – (rating_number, pct) for lower rtg prior to solving for risk score, force this on pool

Returns length of hypo pool, array of ratings after applying force high, force low ratings

Return type numpy.ndarray

model_clo_insight.engines.post_defaults module

Post-Default Pooling Analytics

This module contains the methods used in the analysis after the binary default / non-default process

calc_def_pct_from_amort(*loans*, *ttd_matrix*)

Computes the Exposure at Default Using Transaction Tenor Driven Amortization Table

Per trial (row), compute the percentage of the security that defaults via the security's amort schedule and the simulated time-to-default.

Parameters

- **loans** (*list[clo_internal_objects.CorpLoan]*) – List of loans in the pool
- **ttd_matrix** (*numpy.ndarray*) – n_{trials} x $n_{obligors}$ matrix of simulated default times from *model_clo_insight.engines.defaults.run_defaults*

Returns n_{trials} x $n_{obligors}$ matrix of Exposure-at-Default

Return type np.ndarray

calc_def_pct_from_asset_specific_bullet(*loans*, *ttd_matrix*)
Computes the Exposure at Default Using Asset Specific Tenor Bullet Amortization

Per trial (row), determine if the security defaults via the tenor of the security and its simulated time-to-default

Parameters

- **loans** (*list[clo_internal_objects.CorpLoan]*) – List of loans in the pool
- **ttd_matrix** (*numpy.ndarray*) – n_{trials} x $n_{obligors}$ matrix of simulated default times from *model_clo_insight.engines.defaults.run_defaults*

Returns n_{trials} x $n_{obligors}$ matrix of Exposure-at-Default

Return type np.ndarray

calc_wal(*loans*)
Computes the Weighted Average Life

Loop through loan objects to build pool-wide amortization schedule that is based off total par per quarterly period

Parameters **loans** (*list[clo_internal_objects.CorpLoan]*) – List of loans in the pool

Returns weighted average life of list of loans

Return type float

run_defaults_only(*loans*, *def_pct_matrix*)
Computes cumulative defaults at the pool level per trial

Parameters

- **loans** (*list[clo_internal_objects.CorpLoan]*) – List of loans in the pool
- **def_pct_matrix** – n_{trials} x $n_{obligors}$ matrix of simulated default times from calc_def_pct_from_amort()

Returns n_{trials} x $n_{obligors}$ matrix of default percentage

Return type numpy.ndarray

run_losses_only(*loans*, *def_pct_matrix*)
Computes cumulative losses at the pool level per trial. Requires asset specific recovery inputs.

Parameters

- **loans** (*list[clo_internal_objects.CorpLoan]*) – List of loans in the pool
- **def_pct_matrix** – n_{trials} x $n_{obligors}$ matrix of simulated default times from calc_def_pct_from_amort()

Returns n_{trials} x $n_{obligors}$ matrix of default percentage

Return type numpy.ndarray

model_clo_insight.engines.settings module

Constants used across clo-insight. Store things like bisection tolerance here. Use this for settings that should not be modified by the user, and to avoid hard coding any numbers

model_clo_insight.engines.solvers module

Module which contains the core optimization methods for the hypo pool generation

monitor_metric(*riskscore_max*, *run_mode*, *shell_pool*, *wal*, *coef_sdr*, *risk_score_tbl*, *IDT_sp*, *WARF_sp*, *force_high*, *force_low*)

Computes the 2 metrics used in the CDO Monitor computation

- BDR: Break Even Default Rate
- SDR: Scenario Default Rate (analagous to the DBRSM CLO Asset Model hurdle rate output)

Parameters

- **riskscore_max** (*float*) – max DBRSM risk score of the pool
- **run_mode** (*str*) – Expected portfolio default rate (EPDR) or WA Rating Factor (SP_WARF)
- **shell_pool** (*list[clo_internal_objects.Obligor]*) – pool where obligor & industry percentages have been previously solved for a given Dscore
- **wal** (*float*) – Target portfolio weighted average life
- **coef_sdr** (*list*) – Scenario even default rate (SDR) coefficients for the transaction
- **risk_score_tbl** (*dict[int, float]*) – lookup table for DBRSM Risk Score
- **IDT_sp** (*ndarray*) – idealized default 2-D table; sourced from transaction documents
- **WARF_sp** (*ndarray*) – WARF-equivalent 1-D table; sourced from transaction documents
- **force_high** (*tuple*) – (rating_number, pct) for higher rtg prior to solving for risk score, force this on pool
- **force_low** (*tuple*) – (rating_number, pct) for lower rtg prior to solving for risk score, force this on pool

Returns Monitor SDR, SDR diagnostics dictionary

Return type tuple

monitor_objective_function(*riskscore_max*, *run_mode*, *target_sdr*, *shell_pool*, *wal*, *coef_sdr*, *risk_score_tbl*, *IDT_sp*, *WARF_sp*, *force_high*, *force_low*)

Used as the objective funtion in the optimizers

Parameters

- **riskscore_max** (*float*) – max DBRSM risk score of the pool; this is what gets iterated over when monitor_objective_function is hooked up to the optimizer
- **target_sdr** (*float*) – target Monitor SDR metric that the optimizer attempts to reach by iterating over the
- **shell_pool** (*list[clo_internal_objects.Obligor]*) – pool where obligor & industry percentages have been previously solved for a given Dscore
- **wal** (*float*) – Target portfolio weighted average life

- **coef_sdr** (*list*) – Scenario even default rate (SDR) coefficients for the transaction
- **risk_score_tbl** (*dict[int, float]*) – lookup table for DBRSM Risk Score
- **IDT_sp** (*ndarray*) – idealized default 2-D table; sourced from transaction documents
- **WARF_sp** (*ndarray*) – WARF-equivalent 1-D table; sourced from transaction documents
- **force_high** (*tuple*) – (rating_number, pct) for higher rtg prior to solving for risk score, force this on pool
- **force_low** (*tuple*) – (rating_number, pct) for lower rtg prior to solving for risk score, force this on pool

Returns difference between the actual dscore and the target, solver attempts to minimize this

Return type float

monitor_pool(*riskscore_max*, *shell_pool*, *wal*, *risk_score_tbl*, *IDT_sp*, *force_high*, *force_low*, *is_compute_pd=True*)

Applies rating mixture to pool based on riskcore_max, then PD based on lookup tables. Need this to compute expected PD and PD dispersion metrics for the monitor calculation

Parameters

- **riskscore_max** (*float*) – max DBRSM risk score of the pool
- **shell_pool** (*list[clo_internal_objects.Obligor]*) – pool where obligor & industry percentages have been previously solved for a given Dscore
- **wal** (*float*) – Target portfolio weighted average life
- **risk_score_tbl** (*dict[int, float]*) – lookup table for DBRSM Risk Score
- **IDT_sp** (*ndarray*) – idealized default 2-D table; sourced from transaction documents
- **force_high** (*tuple*) – (rating_number, pct) for higher rtg prior to solving for risk score, force this on pool
- **force_low** (*tuple*) – (rating_number, pct) for lower rtg prior to solving for risk score, force this on pool
- **is_compute_pd** (*bool*) – if True, compute the PD for each obligor via IDT lookup

Returns hypo pool w/ PD overlay

Return type list

solve_dscore_ob_base(*dscore_min*, *ob_base*, *ind_base*, *riskscore_max*, *ob_exceptions*, *ind_exceptions*, *ob_floor*, *ind_floor*, *risk_score_tbl*, *tables*, *force_high*, *force_low*, *is_simple_industry*, *n_simple_industry*)

Solver for a pool based on dscore.

Solves for a pool dscore by forcing ob & ind conc limit exceptions, spinning down the base ob & ind exceptions until the dscore is in compliance. Should not be run directly, but from hypo_pool_master etc.

General Algorithm 1) Construct initial pool using build_pool and calculate dscore 2) If actual dscore > dscore limit, pool is compliant so no need to solve 3) Check lower bound ob base ob/ind max to ensure the solver can find a solution 4) Run numpy's bisection algo by changing the base ob/ind max at a constant ratio (their initial ratio)

Parameters

- **dscore_min** (*float*) – maximum diversity score limit for the pool
- **ob_base** (*float*) – base max obligor concentration limitation

- **ind_base** (*float*) – base max industry concentration limitation
- **riskscore_max** (*float*) – max DBRSM risk score of the pool
- **ob_exceptions** (*list[objects.ConcLimitException]*) – obligor concentration limitations
- **ind_exceptions** (*list[objects.ConcLimitException]*) – industry concentration limitations
- **ob_floor** (*float*) – lowest level that the base obligor concentration limitation is allowed to spin down to
- **ind_floor** (*float*) – lowest level that the base industry concentration limitation is allowed to spin down to
- **risk_score_tbl** (*dict[int, float]*) – lookup table for DBRSM Risk Score
- **tables** (*dict[str, np.ndarray]*) – Lookup tables for Dscore and industry codes
- **force_high** (*tuple*) – high rating (rating integer, percentage) to lock prior to solving for belly ratings (ratings not in the front end or in the back end of the yield curve)
- **force_low** (*tuple*) – low rating (rating integer, percentage) to lock prior to solving for belly ratings (ratings not in the front end or in the back end of the yield curve)
- **is_simple_industry** (*bool*) – if true, industry allocation is done on a repeating sequence. Hypo only iterates on obligor in this case.
- **n_simple_industry** (*int*) – if is_simple_industry is True, this is the number of industries to repeat in sequence

Returns results dictionary

Return type dict

Results Dictionary Keys:

- is_solve: True if solver found solution, False otherwise
- dscore: diversity score calculated on the actual pool
- ob_base: base obligor concentration limitation that the solver found
- ind_base: base industry concentration limitation that the solver found
- obligors: OrderedDict[objects.Obligor] pool of Obligors.

solve_dscore_ob_ex_partial(*dscore_min*, *ob_base*, *ind_base*, *riskscore_max*, *ob_exceptions*, *ind_exceptions*, *ob_ex_partial*, *ob_floor*, *risk_score_tbl*, *tables*, *force_high*, *force_low*, *is_simple_industry*, *n_simple_industry*)

Used when an obligor exception is removed in the outer layer algorithm. This solves for a fractional amount of that exception to be included back in the pool such the the target dscore is met

- Iterates over the smallest obligor concentration limitation exception

Parameters

- **dscore_min** (*float*) – target diversity score for the pool
- **ob_base** (*float*) – base max obligor concentration limitation
- **ind_base** (*float*) – base max industry concentration limitation
- **riskscore_max** (*float*) – max DBRSM risk score of the pool

- **ob_exceptions** (*list[objects.ConcLimitException]*) – obligor concentration limitations
- **ind_exceptions** (*list[objects.ConcLimitException]*) – industry concentration limitations
- **ob_ex_partial** (*float*) – the current smalled obligor concentration limitation exception; gets controlled by solver
- **ob_floor** (*float*) – lowest level that the base obligor concentration limitation is allowed to spin down to
- **risk_score_tbl** (*dict[int, float]*) – lookup table for DBRSM Risk Score
- **tables** (*dict[str, np.ndarray]*) – Lookup tables for Dscore and industry codes
- **force_high** (*tuple*) – high rating (rating integer, percentage) to lock prior to solving for belly ratings (ratings not in the front end or in the back end of the yield curve)
- **force_low** (*tuple*) – low rating (rating integer, percentage) to lock prior to solving for belly ratings (ratings not in the front end or in the back end of the yield curve)
- **is_simple_industry** (*bool*) – if true, industry allocation is done on a repeating sequence. Hypo only iterates on obligor in this case.
- **n_simple_industry** (*int*) – if is_simple_industry is True, this is the number of industries to repeat in sequence

Returns results dictionary

Return type dict

solver_wrap_base_ob(*ob_base*, *ind_mult*, *target_dscore*, *riskscore_max*, *ob_exceptions*, *ind_exceptions*, *ind_floor*, *risk_score_tbl*, *tables*, *force_high*, *force_low*, *is_simple_industry*, *n_simple_industry*)

Objective function passed to the solver algorithm

- Iterates over base obligor concentration limitation
- Base industry concentration expressed as a multiple of base obligor

Parameters

- **ob_base** (*float*) – base max obligor concentration limitation, this gets controlled by the solver
- **ind_mult** (*float*) – multiplier to reduce the base max ind by a constant proportion to the base max ob
- **target_dscore** (*float*) – target diversity score limit for the pool
- **riskscore_max** (*float*) – max DBRSM risk score of the pool
- **ob_exceptions** (*list[objects.ConcLimitException]*) – obligor concentration limitations
- **ind_exceptions** (*list[objects.ConcLimitException]*) – industry concentration limitations
- **ind_floor** (*float*) – minimum that the base industry pct can go; overrides the ind_mult * ob_max if that is too low
- **risk_score_tbl** (*dict[int, float]*) – lookup table for DBRSM Risk Score
- **tables** (*dict[str, np.ndarray]*) – Lookup tables for Dscore and industry codes

- **force_high** (*tuple*) – high rating (rating integer, percentage) to lock prior to solving for belly ratings (ratings not in the front end or in the back end of the yield curve)
- **force_low** (*tuple*) – low rating (rating integer, percentage) to lock prior to solving for belly ratings (ratings not in the front end or in the back end of the yield curve)
- **is_simple_industry** (*bool*) – if true, industry allocation is done on a repeating sequence. Hypo only iterates on obligor in this case.
- **n_simple_industry** (*int*) – if is_simple_industry is True, this is the number of industries to repeat in sequence

Returns difference between the actual dscore and the target, solver attempts to minimize this

Return type float

solver_wrap_ob_ex_partial(*ob_ex_partial*, *ob_base*, *ind_base*, *target_dscore*, *riskscore_max*, *ob_exceptions*, *ind_exceptions*, *risk_score_tbl*, *tables*, *force_high*, *force_low*, *is_simple_industry*, *n_simple_industry*)

Objective function passed to the solver algorithm

- Iterates over smallest obligor concentration exception limitation percentage

Parameters

- **ob_ex_partial** (*float*) – the current smallest obligor concentration limitation exception; gets controlled by solver
- **ob_base** (*float*) – base max obligor concentration limitation,
- **ob_base** – base max industry concentration limitation
- **target_dscore** (*float*) – target diversity score limit for the pool
- **riskscore_max** (*float*) – max DBRSM risk score of the pool
- **ob_exceptions** (*list[clo_internal_objects.ConcLimitException]*) – obligor concentration limitations
- **ind_exceptions** (*list[clo_internal_objects.ConcLimitException]*) – industry concentration limitations
- **risk_score_tbl** (*dict[int, float]*) – lookup table for DBRSM Risk Score
- **tables** (*dict[str, np.ndarray]*) – Lookup tables for Dscore and industry codes
- **force_high** (*tuple*) – high rating (rating integer, percentage) to lock prior to solving for belly ratings (ratings not in the front end or in the back end of the yield curve)
- **force_low** (*tuple*) – low rating (rating integer, percentage) to lock prior to solving for belly ratings (ratings not in the front end or in the back end of the yield curve)
- **is_simple_industry** (*bool*) – if true, industry allocation is done on a repeating sequence. Hypo only iterates on obligor in this case.
- **n_simple_industry** (*int*) – if is_simple_industry is True, this is the number of industries to repeat in sequence

Returns difference between the actual dscore and the target, solver attempts to minimize this

Return type float

4.1.3 model_clo_insight.methodology_assumptions package

4.1.3.1 Submodules

model_clo_insight.methodology_assumptions.meth_param_loader module

model_clo_insight.methodology_assumptions.methodology_ref module

import_json(*file_name*)
Import Json data from files

Parameters **file_name** (*string*) – file name

Returns JSON data

Return type dict

import_json_part(*file_name*, *key*)
Import parts of json files to the data

Parameters

- **file_name** (*string*) – file name
- **key** (*string*) – dictionary key for lookup

Returns JSON data

Return type dict

tbl_to_json(*file_name*)
Import Tabular Data from csv files & Json conversion function

Parameters **file_name** (*string*) – file name

Returns JSON data

Return type dict

4.2 Submodules

4.2.1 model_clo_insight.app module

Module for the Methods to be Hosted as RESTful APIs

- Can be accessed as RESTful APIs with web-based clients such as Jupyter Notebooks or Javascript UIs
- Can be accessed locally via XL Wings

_step_5_final_results_mode_1_2(*deal*, *inputs*)
Performs the final aggregation of trading scenario results from 4a (cashflow BDRs, pasted in from Proprietary Cashflow Engine) and the Predictive Model simulation results matrix. The cushion per scenario is calculated, and a summary matrix computed. This is the final step in the analysis.

For modes 1 & 2, we use the Scenario Matrix (generated in step 3) as the base matrix for merging in all the results

Parameters

- **deal** (*CLOTransaction*) – transaction capital structure and key terms

- **inputs** (*InputStep5*) – details in *model_clo_insight.clo_external_data_contracts.InputStep5*

Return type *OutputStep5*

Returns details in *model_clo_insight.clo_external_data_contracts.OutputStep5*

General Steps

1) Build tranche and scenario dataframes
2) Determine default hurdle rates for each tranche
3) Build dataframe of break evens (BDRs) that were pasted in from Proprietary Cashflow Engine
4) Determine governing BDRs per tranche per scenario (min for AAA, avg for the rest)
5) Compute cushion (BDR - default hurdle rate)
6) Filter scenarios by the user 'is_on' choice
7) Compute stats and summary results table
8) Setup output dataclass structure

_step_5_final_results_mode_3(*deal*, *inputs*)

Performs the final aggregation of trading scenario results from 4a (cashflow BDRs, pasted in from Proprietary Cashflow Engine) and the Predictive Model simulation results matrix. The cushion per scenario is calculated, and a summary matrix computed. This is the final step in the analysis.

For Mode 3, we use the deal matrix as the base matrix for merging in all the results

Parameters

- **deal** (*CLOTransaction*) – transaction capital structure and key terms
- **inputs** (*InputStep5*) – details in *model_clo_insight.clo_external_data_contracts.InputStep5*

Return type *OutputStep5*

Returns details in *model_clo_insight.clo_external_data_contracts.OutputStep5*

General Steps

1) Build tranche and deal matrix dataframes
2) Build scenario dataframe
3) Map default hurdle rates from scenario matrix -> deal matrix
4) Determine default hurdle rates for each tranche
5) Build dataframe of break evens (BDRs) that were pasted in from Proprietary Cashflow Engine
6) Determine governing BDRs per tranche per scenario (min for AAA, avg for the rest)
7) Compute cushion (BDR - default hurdle rate)
8) Compute stats and summary results table
9) Setup output dataclass structure

step_1_parse_pool(*deal*, *indicative_pool*, *overrides=None*)

Ingests the transaction structure and indicative portfolio. Computes key metrics on the indicative portfolio. Loads in assumption tables from methodology reference. Note that outputs from this step are used in multiple subsequent steps

Parameters

- **deal** (*CLOTransaction*) – transaction capital structure and key terms like deal.trading_scenario_mode
- **indicative_pool** (List[*CLOPoolRow*]) – portfolio data (loan tape)
- **overrides** (Optional[Dict]) – optional input to override methodology parameters

Return type *OutputStep1*

Returns pool metrics, modeling pool, and recommended values for trading scenarios

General Steps

1) Indicative portfolio: Mapped data
2) Indicative portfolio: Derived data (computed values)
3) Compute pool metrics
4) Compute portfolio strats
5) Build replines - Indicative Portfolio
6) Build replines - Trading scenario recommendations
7) Compute trading scenario recommended parameters
8) Rounding and date cleanup
9) Setup output dataclass structure

step_2_run_indicative_pool(*deal*, *inputs*, *overrides=None*)

Ingests the indicative portfolio and cutpoint tenors. Runs the core Predictive Model algorithm to generate the cumulative default distribution & compute percentiles -> hurdle rates

Parameters

- **deal** (*CLOTransaction*) – transaction capital structure and key terms
- **inputs** (*InputStep2*) – details in *model_clo_insight.clo_external_data_contracts.InputStep2*
- **overrides** (Optional[Dict]) – optional input to override methodology parameters

Return type *OutputStep2*

Returns details in *model_clo_insight.clo_external_data_contracts.OutputStep2*

General Steps

1) Create internal CLO objects (obligors, loans, transaction, settings, meth params)
2) Setup object references
3) Run simulation to generate time-to-default (ttd) matrix
4) Loop through each rating-based amortization schedule to compute default hurdle rates and diagnostic statistics (this is how prepayment assumptions are implemented)
5) Create tranche results dataframe
6) Setup output dataclass structure

step_3_build_trading_scenarios(*deal*, *inputs*, *overrides=None*)

Builds out trading scenarios as per new methodology approach. Runs hypo pool on each to infer WARF/risk score, but do not run simulation algo.

Recommended inputs in this step are displayed to the Analyst as Step 1 output, but the analyst may choose to use the recommendend values or override the recommended values using the choice values. If analyst chooses to override recommendend values, they must include a rationale in the comment fields.

Parameters

- **deal** (*CLOTransaction*) – transaction capital structure and key terms
- **inputs** (*InputStep3*) – details in *model_clo_insight.clo_external_data_contracts.InputStep3*
- **overrides** (Optional[Dict]) – optional input to override methodology parameters

Return type *OutputStep3*

Returns details in *model_clo_insight.clo_external_data_contracts.OutputStep3*

General Steps

1) If mode 3, map deal matrix to modeling matrix (row for row) and create dataframe
2) If mode 1 or 2, create dataframe for investment grade scenarios
3) If mode 1 or 2, create dataframe for AAA scenarios
4) Setup hypo pool parameters and lookup tables
5) Main loop:
   a) Solve for industry/obligor concentration limitations s.t. target Dscore is realized
   b) If mode 1, solve for rating mixture s.t. BDR ~ SDR (e.g. the CDO Monitor test), For Mode 2 and 3, the respective deal matrices are used.
   c) Build a row for the results, and the diagnostics
6) Build results and diagnostics dataframes
7) Setup output dataclass structure

step_4a_generate_cf_scenarios(*deal*, *inputs*, *overrides=None*)

This step is used to build out the cash flow matrix for the trading scenarios. It is designed to be used in Proprietary Cashflow Engine out-of-the-box, but may also be used in Intex with some modifications. The scenarios are built around the unique combinations of WAS and seniority/WARR that arise from the trading scenarios.

Parameters

- **deal** (*CLOTransaction*) – transaction capital structure and key terms
- **inputs** (*InputStep4a*) – details in *model_clo_insight.clo_external_data_contracts.InputStep4a*
- **overrides** (Optional[Dict]) – optional input to override methodology parameters

Return type *OutputStep4a*

Returns details in *model_clo_insight.clo_external_data_contracts.OutputStep4a*

General Steps

1) If mode 3:
   a) Determine unique WAS
   b) Determine unique WARR and create an integer ID
   c) Determine repline balances s.t. target WARR is met for all unique WARR keys
2) If mode 1, 2:

a) Create filtered scenario dataframe based on analyst 'is_on' choice

b) Create integer keys for seniority

3) Create integer keys for WAS

4) Use numpy operations to create scenario matrix

5) Setup output dataclass structure

step_4b_run_trading_scenarios(*deal*, *inputs*, *overrides=None*)

This method runs the trading scenario matrix through the Predictive Model simulation. The hypo pools are re-generated identical to step 3, and the resulting pools run through the simulation.

Parameters

- **deal** (*CLOTransaction*) – transaction capital structure and key terms
- **inputs** (*InputStep4b*) – details in *model_clo_insight.clo_external_data_contracts.InputStep4b*
- **overrides** (Optional[Dict]) – optional input to override methodology parameters

Return type *OutputStep4b*

Returns details in *model_clo_insight.clo_external_data_contracts.OutputStep4b*

General Steps

1) Setup scenario dataframe and filter out based on user 'is_on' choice

2) Setup hypo pool parameters and lookup tables

3) Create internal CLO objects (transaction, settings, meth params)

4) Setup amortization matrix

5) Main execution loop:

a) Solve for industry/obligor concentration limitations s.t. target Dscore is realized

b) If mode 1, solve for rating mixture s.t. BDR ~ SDR (e.g. the CDO Monitor test). For mode 2 and 3, use hypo_pool.HypoResultMonitor to generate results using matrices.

c) Create lists of Obligor and Loan objects from the hypo pool

d) Run simulation to generate time-to-default (ttd) matrix

e) Loop through each rating-based amortization schedule to compute default hurdle rates and diagnostic statistics (this is how prepayment assumptions are implemented)

6) Create results and diagnostics dataframes

7) setup output dataclass structure

step_5_final_results(*deal*, *inputs*, *overrides=None*)

Wrapper for the final aggregation method. The logic is quite different for modes 1, 2 vs 3. There are 2 internal methods below that handle each of these paths:

- *model_clo_insight.app._step_5_final_results_mode_1_2*
- *model_clo_insight.app._step_5_final_results_mode_3*

Parameters

- **deal** (*CLOTransaction*) – transaction capital structure and key terms

- **inputs** (*InputStep5*) – details in *model_clo_insight.clo_external_data_contracts.InputStep5*
- **overrides** (Optional[Dict]) – optional input to override methodology parameters

Return type *OutputStep5*

Returns details in *model_clo_insight.clo_external_data_contracts.OutputStep5*

4.2.2 model_clo_insight.clo_external_data_contracts module

This module serves as the location for all of the external model object classes used to communicate with Rest APIs or Excel.

class AgencyRecovery(*seniority=None*, *rec_rate=None*, ***kwargs*)

Bases: object

Used to store non-DBRSM rating agency recovery assumptions (by seniority only, no country tiering)

rec_rate: float = None
other agency's recovery rate assumption for this seniority level (estimated from historical analysis)

seniority: str = None
seniority level (sr secured, cov-lite, second lien)

class CFMatrixMetaData(*tranche=None*, *is_retranche=None*, *dbrs_rtg=None*, *libor_curve=None*, *def_pattern=None*, ***kwargs*)

Bases: object

Table row that represents the inputs to a Proprietary Cashflow Engine BDR scenario

dbrs_rtg: str = None
DBRSM rating stress level

def_pattern: str = None
default timing curve

is_retranche: int = None
not used

libor_curve: int = None
interest rate curve

tranche: str = None
which tranche the BDR is computed on

class CFNamedRangeRow(*named_range=None*, *row=None*, *col=None*, ***kwargs*)

Bases: object

Table row representing the Excel named ranges that are controlled by Proprietary Cashflow Engine's dynamic runner

col: int = None
column of the named range

named_range: str = None
Excel named range

row: int = None
row of the named range

class CFRepline(*name=None*, *dbrs_tier=None*, *par=None*, *percent=None*, *cpn_type_int=None*, *daycount=None*, *spread=None*, *coupon=None*, *rec_lag=None*, ***kwargs*)

Bases: object

Data structure for a cash flow analysis repline (representative group of collateral loans)

coupon: float = None
fixed coupon

cpn_type_int: float = None
fixed vs float

daycount: str = None
30/360 for fixed and act/360 for float

dbrs_tier: int = None
DBRSM country tier (effects recovery rate and lag)

name: str = None
name of the repline

par: float = None
aggregate par amount

percent: float = None
this repline's percentage of the total collateral

rec_lag: int = None
recovery lag (expressed as number of deal payment periods)

spread: float = None
floating spread

class CFReplineTradingPct(*dbrs_tier=None*, *cpn_type_int=None*, *daycount=None*, *rec_lag=None*, *name=None*, *s_1=None*, *s_2=None*, *s_3=None*, ***kwargs*)

Bases: object

Data structure used for storing the information about each seniority/tier combination used in the trading scenario analysis

cpn_type_int: float = None
fixed vs float

daycount: str = None
daycount value

dbrs_tier: int = None
DBRSM country tier

name: str = None
unique identifier

rec_lag: int = None
Recovery lag based on country tier

s_1: float = None
percentage allocation in seniority regime 1 (all senior secured only)

s_2: float = None
percentage allocation in seniority regime 2 (max cov-lite)

s_3: float = None
percentage allocation in seniority regime 3 (max cov-lite & max second lien)

class CLOConcLimitException(*num=None*, *pct=None*, ***kwargs*)
Bases: object

Represents a concentration limitation exception (used for obligor and industry)

num: int
number of exceptions allowed at this level

pct: float
concentration limitation level for a set number of exceptions

class CLOFee(*floating=None*, *fixed=None*, ***kwargs*)
Bases: object

Represents a fee due by the CLO

fixed: float
fixed component of the fee

floating: float
floating component of the fee

class CLOPoolRow(*obligor=None*, *security=None*, *ind_dbrsm=None*, *country=None*, *seniority=None*, *is_cov_lite=None*, *is_defaulted=None*, *cpn_type=None*, *cpn=None*, *spd=None*, *libor_floor=None*, *dt_maturity=None*, *rtg_mdy=None*, *rtg_sp=None*, *rtg_fitch=None*, *rtg_dbrs=None*, *rtg_dbrs_ce=None*, *rec_rate_mdy=None*, *rec_rate_sp=None*, *rtg_watch_mdy=None*, *rtg_watch_sp=None*, *rtg_watch_fitch=None*, *rtg_watch_dbrs=None*, *par=None*, *unfunded=None*, *ind_code_dbrsm=None*, ***kwargs*)
Bases: object

Represents a single row for the Indicative Portfolio

country: str = None
DBRSM country code. Set at the obligor level (first occurrence)

cpn: float = None
Fixed coupon. Used in repline construction

cpn_type: str = None
Fixed vs floating. Used in repline construction

dt_maturity: datetime.datetime = None
Legal maturity date for the security

ind_code_dbrsm: int = None
DBRSM industry code. Set at the obligor level (first occurrence)

ind_dbrsm: str = None
DBRSM industry name. Set at the obligor level (first occurrence)

is_cov_lite: str = None
True if loan is a cov-lite loan (e.g. lack of maintenence covenants). Drives recovery assumptions.

is_defaulted: str = None
True if loan is defaulted loan. Exclude defaults from asset default simulations, and included from repline builder for Proprietary Cashflow Engine

libor_floor: float = None
LIBOR floor (if applicable). Currently not used in the analysis

obligor: str = None
unique obligor ID. Obligors with the same string ID share the same random vars in the simulation

par: float = None
Par amount of the security

rec_rate_mdy: float = None
Moody's recovery rate. Determined by difference in corp family vs facility rating

rec_rate_sp: float = None
S&P recovery rate (not recovery rating!).

rtg_dbrs: str = None
DBRSM credit rating. Set at the obligor level (first occurrence)

rtg_dbrs_ce: str = None
DBRSM credit estimate. Set at the obligor level (first occurrence)

rtg_fitch: str = None
Fitch Issuer credit rating. Set at the obligor level (first occurrence)

rtg_mdy: str = None
Moodys obligor default probability or corporate family credit rating. Set at the obligor level (first occurrence)

rtg_sp: str = None
S&P Issuer credit rating. Set at the obligor level (first occurrence)

rtg_watch_dbrs: str = None
DBRSM credit watch. Set at the obligor level (first occurrence)

rtg_watch_fitch: str = None
Fitch credit watch. Set at the obligor level (first occurrence)

rtg_watch_mdy: str = None
S&P credit watch. Set at the obligor level (first occurrence)

rtg_watch_sp: str = None
Moodys credit watch. Set at the obligor level (first occurrence)

security: str = None
unique security ID. Two line items with same obligor ID should have different security ID

seniority: str = None
Seniority level (e.g. senior secured vs second lien). Set at the security level

spd: float = None
Floating spread. Used in repline construction and WAS calculation

unfunded: float = None
remaining undrawn amount (if revolver or delayed draw loan)

class CLOTranche(*name=None*, *bond_group=None*, *par=None*, *unfunded=None*, *assumed_draw=None*, *cpn_type=None*, *cpn=None*, *spd=None*, *rtg_sp=None*, *rtg_mdy=None*, *rtg_fitch=None*, *rtg_dbrs=None*, *oc_test=None*, *ic_test=None*, *rvst_oc_test=None*, ***kwargs*)

Bases: object

Represents a tranche of a CLO's capital structure

assumed_draw: float = None
amount to assume is drawn when running the analysis

bond_group: int = None
integer representing the paydown order under the note payment sequence

cpn: float = None
fixed coupon

cpn_type: str = None
fixed vs floating

ic_test: float = None
Interest coverage test associated with this tranche's bond group

name: str = None
name of the tranche

oc_test: float = None
Overcollateralization test associated with this tranche's bond group

par: float = None
par amount

rtg_dbrs: str = None
DBRSM rating (proposed)

rtg_fitch: str = None
Fitch rating

rtg_mdy: str = None
Moody's rating

rtg_sp: str = None
S&P rating

rvst_oc_test: float = None
Reinvestment overcollateralization test associated with this tranche's bond group

spd: float = None
floating spread

unfunded: float = None
undrawn amount (for revolvers and delayed draws)

class CLOTrancheOutputStep1(*name=None*, *bond_group=None*, *par=None*, *rtg_dbrs=None*, *par_subordination=None*, ***kwargs*)

Bases: object

Represents a tranche row in the tranche-related input for step 2

bond_group: int = None
integer representing the paydown order under the note payment sequence

name: str = None
name of the tranche

par: float = None
par amount

par_subordination: float = None
par subordination for this tranche (computed based on collateral par and total par of tranches senior to this tranche

rtg_dbrs: str = None
DBRSM rating (proposed)

class CLOTrancheOutputStep2(*name=None*, *bond_group=None*, *par=None*, *rtg_dbrs=None*, *par_subordination=None*, *davinci_tranche_name=None*, *exp_maturity=None*, *BDR=None*, *default_hurdle=None*, *cushion=None*, *result=None*, ***kwargs*)

Bases: object

Represents a tranche row in the tranche-related output of step 2

BDR: float = None
break even default rate (output from Proprietary Cashflow Engine)

bond_group: int = None
integer representing the paydown order under the note payment sequence

cushion: float = None
BDR - default hurdle

davinci_tranche_name: str = None
name of the tranche in Proprietary Cashflow Engine (used for lookups later)

default_hurdle: float = None
cumulative default hurdle rate from the predictive model simulation (lookup via propsosed DBRSM rating)

exp_maturity: float = None
expected bond maturity (output from Proprietary Cashflow Engine)

name: str = None
name of the tranche

par: float = None
par amount

par_subordination: float = None
par subordination for this tranche (computed based on collateral par and total par of tranches senior to this tranche

result: float = None
pass if cushion >= 0, fail if < 0

rtg_dbrs: str = None
DBRSM rating (proposed)

class CLOTrancheOutputStep5(*name=None*, *davinci_tranche_name=None*, *bond_group=None*, *par=None*, *rtg_dbrs=None*, *par_subordination=None*, *bdr_min=None*, *hurdle_min=None*, *min=None*, *n_fail=None*, *min_loc=None*, *is_pass=None*, ***kwargs*)

Bases: object

Represents a tranche row in the tranche-related output of step 5

Note: inheritance not used here due as it changes in order of the fields

bdr_min: float = None
BDR for the min cushion

bond_group: int = None
integer representing the paydown order under the note payment sequence

davinci_tranche_name: str = None
name of the tranche in Proprietary Cashflow Engine (used for lookups later)

hurdle_min: float = None
hurdle for the min cushion

is_pass: int = None
True if the tranche passes all scenarios

min: float = None
minimum cushion

min_loc: int = None
scenario number with the smallest cushion

n_fail: int = None
number of trading scenarios where cushion < 0

name: str = None
name of the tranche

par: float = None
par amount

par_subordination: float = None
par subordination for this tranche (computed based on collateral par and total par of tranches senior to this tranche

rtg_dbrs: str = None
DBRSM rating (proposed)

class CLOTransaction(*dt_closing=None*, *dt_first_pay=None*, *dt_end_rvst=None*, *dt_non_call=None*, *dt_maturity=None*, *pmt_per_yr=None*, *rvst_target_par=None*, *total_capitalization=None*, *max_wal=None*, *max_warf=None*, *mdy_warr_floor=None*, *mdy_warr_cap=None*, *trading_scenario_mode=None*, *is_surv_mode=None*, *dt_last_pmt=None*, *libor_spot=None*, *max_wal_surv=None*, *princ_proceeds=None*, *ce_in_transit=None*, *fee_sr_admin=None*, *fee_sr_mgmt=None*, *fee_jr_mgmt=None*, *fee_incentive=None*, *cl_obligor=None*, *cl_industry=None*, *cl_cov_lite=None*, *cl_second_lien=None*, *cl_fixed=None*, *cl_long_dated=None*, *cl_dbrs_II=None*, *cl_dbrs_III=None*, *cl_dbrs_IV=None*, *cl_ex_obligor=None*, *cl_ex_industry=None*, *tranches=None*, *revolver_method=None*, *net_agg_exp_amt=None*, *perf_par_adj_amt=None*, *perf_par_adj_rationale=None*, *mdy_warr_cushion_floor=None*, ***kwargs*)

Bases: object

Top level data structure that represents the transaction level data for a CLO

ce_in_transit: float = None
Credit Estimate in-transit value (used for obligors with missing CE)

cl_cov_lite: float = None
max cov-lite loans

Type concentration limitation

cl_dbrs_II: float = None
max DBRSM Tier II countries

Type concentration limitation

cl_dbrs_III: float = None
max DBRSM Tier III countries

Type concentration limitation

cl_ex_industry:
List[*model_clo_insight.clo_external_data_contracts.CLOConcLimitException*] = None
instances of *CLOConcLimitException* that represent the industry concentration limitations

cl_ex_obligor:
List[*model_clo_insight.clo_external_data_contracts.CLOConcLimitException*] = None
instances of *CLOConcLimitException* that represent the obligor concentration limitations

cl_fixed: float = None
max fixed rate

Type concentration limitation

cl_industry: float = None
max single obligor

Type concentration limitation

cl_long_dated: float = None
max long dated

Type concentration limitation

cl_obligor: float = None
max single obligor

Type concentration limitation

cl_second_lien: float = None
max second lien loans

Type concentration limitation

dt_closing: datetime.datetime = None
closing date

dt_end_rvst: datetime.datetime = None
end of the reinvestment period

dt_first_pay: datetime.datetime = None
first payment date

dt_last_pmt: datetime.datetime = None
last payment date (used to calculate remaining reinvestment period)

dt_maturity: datetime.datetime = None
legal maturity date

dt_non_call: datetime.datetime = None
end of the non-call period

fee_incentive: *model_clo_insight.clo_external_data_contracts.CLOFee* = None
instance of *CLOFee* - incentive fee

fee_jr_mgmt: *model_clo_insight.clo_external_data_contracts.CLOFee* = None
instance of *CLOFee* - junior management fee

fee_sr_admin: *model_clo_insight.clo_external_data_contracts.CLOFee* = None
instance of *CLOFee* - senior administrative fee

fee_sr_mgmt: *model_clo_insight.clo_external_data_contracts.CLOFee* = None
instance of *CLOFee* - senior management fee

is_surv_mode: str = None
surveillance mode (True for Surveillance, False for New Issue)

libor_spot: float = None
spot libor/sofr as of the closing date

max_wal: float = None
Maximum weighted average life (WAL) test

max_wal_surv: float = None
Maximum weighted average life (WAL) test as of surveillance date

max_warf: float = None
Maximum weighted average rating factor test (global constraint in addition to CDO monitor / collateral quality matrix)

mdy_warr_cap: float = None
Moody's WARR cap for use in modifier matrix

mdy_warr_cushion_floor: float = None
Moody's WARR Cushion floor

mdy_warr_floor: float = None
Moody's WARR floor for use in modifier matrix

net_agg_exp_amt: float = None
Net Aggregate Exposure Amount (total unfunded collateral par - principal proceeds

perf_par_adj_amt: float = None
Analyst adjustment to performing par (+ or - depending on how the trustee calcs work)

perf_par_adj_rationale: str = None
Analyst comment on why they are using the perf par adjustment

pmt_per_yr: int = None
payments per year

princ_proceeds: float = None
pricipal proceeds (repline balances will be adjusted on pro-rata basis as principal proceeds are used for reinvestment purposes during reinvestment period)

revolver_method: int = None
1=use explicit funded vs unfunded in Indicative Portfolio, 2=use Net Aggregate Exposure Amount

rvst_target_par: float = None
Reinvestment Target Par (defined term in the docs, this is the par amount we model to)

total_capitalization: float = None
Optional for certain middle market CLOs where the concentration limitations are computed off this vs the actual par

trading_scenario_mode: float = None
mode 1 for continuous formula constraints, mode 2 for discrete matrix constraints, mode 3 for full DBRSM matrix

tranches: List[*model_clo_insight.clo_external_data_contracts.CLOTranche*] = None
instances of *CLOTranche*

class CorrelationMatrix(*same_reg_same_ind=None*, *same_reg_diff_ind=None*, *diff_reg_same_ind=None*, *diff_reg_diff_ind=None*, ***kwargs*)

Bases: object

correlation matrix dataclass contains 4 values based on a list of lists in Excel, values are taken out and made into 4 separate values

diff_reg_diff_ind: float = None
depricated in methodology, but functionality available in model

diff_reg_same_ind: float = None
depricated in methodology, but functionality available in model

same_reg_diff_ind: float = None
asset correlation between different industries

Type float

same_reg_same_ind: float = None
asset correlation within industry

Type float

class DBRSMatrixInputs(*risk_score=None*, *dscore=None*, *advance_rate=None*, *tenor=None*, *pool_par=None*, *conc_limit_basis=None*, *was=None*, *warr=None*, ***kwargs*)

Bases: object

Table row that represents a row in a DBRSM-style collateral quality matrix

advance_rate: float = None
advance rate if empty we need pool par

conc_limit_basis: float = None
conc limit basis use pool par if empty

dscore: float = None
min diversity score constraint

pool_par: float = None
pool par (varies for ramp-up CLOs)

risk_score: float = None
max DBRSM risk score constraint

tenor: float = None
modeling tenor

warr: float = None
min DBRSM WA recovery rate constraint

was: float = None
min WA spread constraint

class DaVinciResults(*davinci_tranche_name=None*, *exp_maturity=None*, *BDR=None*, ***kwargs*)

Bases: object

BDR: float = None
break even default rate (output from Proprietary Cashflow Engine)

davinci_tranche_name: str = None
name of the tranche in Proprietary Cashflow Engine (used for lookups later)

exp_maturity: float = None
expected bond maturity (output from Proprietary Cashflow Engine)

class DetailedDiagnosticRow

Bases: object

Table row that contains the diagnostic info from the predictive model simulation

class ForceRating(*rtg=None*, *pct=None*, ***kwargs*)

Bases: object

Tuple for storing forced rating assumptions for the hypo pool generator

pct: float = None
percentage to set prior to running the hypo pool solvers

rtg: str = None
rating level

class HypoException

Bases: object

Table row that contains the hypo exception results from step 4b

class InputStep1(*deal=None*, *indicative_port=None*, ***kwargs*)

Bases: object

Master data structure that represents the inputs to *model_clo_insight.app.step_1_parse_pool()*

deal: *model_clo_insight.clo_external_data_contracts.CLOTransaction* = None
instance of *CLOTransaction* that represents the transaction data

indicative_port: List[*model_clo_insight.clo_external_data_contracts.CLOPoolRow*] = None
instances of *CLOPoolRow* that represent the indicative portfolio

class InputStep2(*amort_tail_choice_indicative=None*, *rvst_per_choice_indicative=None*, *tranche_data=None*, *davinci_results=None*, *modeling_portfolio=None*, *modeling_portfolio_asset_specific=None*, *amort_tail_comment_indicative=None*, *rvst_per_comment_indicative=None*, ***kwargs*)

Bases: object

Master data structure that represents the inputs to *model_clo_insight.app.step_2_run_indicative_pool()*

amort_tail_choice_indicative: str = None
user choice for the amortization schedule lookup

amort_tail_comment_indicative: str = None
user comment for choice of amortization schedule

davinci_results: List[*model_clo_insight.clo_external_data_contracts.DaVinciResults*] = None
instances of *DaVinciResults* that represent the Proprietary Cashflow Engine results inputs (BDR, final maturity)

modeling_portfolio:
List[*model_clo_insight.clo_external_data_contracts.ModelPoolRow*] = None
instances of *ModelPoolRow* that represent the modeling portfolio (e.g. ratings & industries mapped)

modeling_portfolio_asset_specific:
List[*model_clo_insight.clo_external_data_contracts.ModelPoolRowAssetSpecific*] = None
modeling portoflio with asset specific input for ABS deals

rvst_per_choice_indicative: int = None
user choice for the number of periods in the cash flow modeling that covers the reinvestment period

rvst_per_comment_indicative: str = None
user comment for choice of number of periods in the cash flow modeling that covers the reinvestment period

tranche_data:
List[*model_clo_insight.clo_external_data_contracts.CLOTrancheOutputStep1*] = None
instances of *CLOTrancheOutputStep1* that represent the tranche data inputs (Par, DBRSM Rating and Par Subordination)

class InputStep3(*hypo_diversity_mode=None*, *hypo_n_simple_ind=None*, *trading_scenario_hypo_mode_comment=None*, *trading_scenario_n_ind_comment=None*, *was_ind_port_choice=None*, *warf_ind_port_choice=None*, *dbrs_risk_score_ind_port_choice=None*, *dscore_ind_port_choice=None*, *was_high_choice=None*, *was_low_choice=None*, *was_low_comment=None*, *dscore_high_choice=None*, *dscore_low_choice=None*, *dscore_high_comment=None*, *dscore_low_comment=None*, *AAA_was_choice=None*, *AAA_was_comment=None*, *repline_trading_pct_choice=None*, *monitor_coef_bdr=None*, *monitor_coef_sdr=None*, *rec_rates_agency_choice=None*, *force_rtg_hi_choice=None*, *force_rtg_lo_choice=None*, *wal_monitor_choice=None*, *max_warf_drift_choice=None*, *deal_matrix_dbrs=None*, *deal_matrix_dbrs_warr_rtg=None*, *deal_matrix_moody_dscore=None*, *deal_matrix_moody_was=None*, *deal_matrix_moody_warf=None*, *deal_matrix_mod_moody_dscore=None*, *deal_matrix_mod_moody_was=None*, *deal_matrix_mod_moody_warf=None*, *repline_trading_pct_comment=None*, *rec_rates_agency_comment=None*, *monitor_coef_bdr_comment=None*, *monitor_coef_sdr_comment=None*, *force_rtg_hi_comment=None*, *force_rtg_lo_comment=None*, *wal_monitor_comment=None*, *max_warf_drift_comment=None*, *deal_matrix_dbrs_warr_rtg_comment=None*, *amort_tail_choice_matrix=None*, *rvst_per_choice_matrix=None*, *amort_tail_comment_matrix=None*, *rvst_per_comment_matrix=None*, *was_ind_port_comment=None*, *warf_ind_port_comment=None*, *dscore_ind_port_comment=None*, ***kwargs*)

Bases: object

Master data structure that represents the inputs to *model_clo_insight.app.step_3_build_trading_scenarios()*

AAA_was_choice: List[float] = None
entire permitted WAS range as per the transaction documents. User chooses how granular to discretize

AAA_was_comment: List[str] = None
comment for AAA WAS choice

amort_tail_choice_matrix: float = None
user choice for the amortization schedule lookup

amort_tail_comment_matrix: str = None
user comments for change of amort tail years

dbrs_risk_score_ind_port_choice: float = None
WA DBRSM risk score of the indicative portfolio

dbrs_risk_score_ind_port_comment: str = None
WA DBRSM risk score of the indicative portfolio

deal_matrix_dbrs:
List[*model_clo_insight.clo_external_data_contracts.DBRSMatrixInputs*] = None
instances of *DBRSMatrixInputs* that represent the transaction's collateral quality matrix if structured to DBRS's methodology (trading_scenario_mode = 3)

deal_matrix_dbrs_warr_rtg: str = None
for deals structured to DBRSM's methodology (trading_scenario_mode = 3), the rating stress level that the WA recovery rate test was sized to

deal_matrix_dbrs_warr_rtg_comment: str = None
comment for which rtg was chosen

deal_matrix_mod_moody_dscore: List[int] = None
dscore values for discrete modifier matrix constraints

deal_matrix_mod_moody_warf: List[List[float]] = None
weighted average rating factor values for discrete modifier matrix constraints

deal_matrix_mod_moody_was: List[float] = None
weighted average spread values for discrete modifier matrix constraints

deal_matrix_moody_dscore: List[int] = None
dscore values for discrete matrix constraints

deal_matrix_moody_warf: List[List[float]] = None
weighted average rating factor values for discrete matrix constraints

deal_matrix_moody_was: List[float] = None
weighted average spread values for discrete matrix constraints

dscore_high_choice: int = None
trading scenario user choice for increase in diversity

dscore_high_comment: str = None
comment for dscore high choice

dscore_ind_port_choice: float = None
diversity score of the indicative portfolio

dscore_ind_port_comment: str = None
diversity score of the indicative portfolio

dscore_low_choice: int = None
trading scenario user choice for decrease in diversity

dscore_low_comment: str = None
comment for dscore low choice

force_rtg_hi_choice: *model_clo_insight.clo_external_data_contracts.ForceRating* = None
instance of *ForceRating* that represents the high rating category to be fixed prior to solving for the rating mixture

force_rtg_hi_comment: List[str] = None
comment for force rating high choice

force_rtg_lo_choice: *model_clo_insight.clo_external_data_contracts.ForceRating* = None
instance of *ForceRating* that represents the low rating category to be fixed prior to solving for the rating mixture

force_rtg_lo_comment: List[str] = None
comment for force rating low choice

hypo_diversity_mode: int = None
determines how the industries are allocated in the hypo pool generator. 1 = multiple of obligor base conc limit, 2 = repeating sequence of N industries

hypo_n_simple_ind: int = None
for hypo_diversity_mode_2, the N industries to repeat

max_warf_drift_choice: float = None
user choice absolute max WARF increase in the trading scenarions (e.g. regardless of monitor/matrix, WARF cannot increase by more than this)

max_warf_drift_comment: str = None
comment for max WARF drift choice

monitor_coef_bdr: List[float] = None
BDR coefficients for the CDO Monitor test. For new transactions, this is determined by analyzing comparable transactions

monitor_coef_bdr_comment: List[str] = None
comments for BDR choices

monitor_coef_sdr: List[float] = None
SDR coefficients for the CDO Monitor test. Typically this does not change deal to deal

monitor_coef_sdr_comment: List[str] = None
comments for SDR choices

rec_rates_agency_choice:
List[*model_clo_insight.clo_external_data_contracts.AgencyRecovery*] = None
user choice for generic non-DBRSM rating agency recovery rate assumptions

rec_rates_agency_comment: List[str] = None
comments for recovery rates choices

repline_trading_pct_choice:
List[*model_clo_insight.clo_external_data_contracts.CFReplineTradingPct*] = None
repline percentages user choice for the trading scenario recommendation

repline_trading_pct_comment: List[str] = None
comment for repline trading pct choice

rvst_per_choice_matrix: int = None
user choice for number of periods in the cash flow modeling that covers the reinvestment period

rvst_per_comment_matrix: str = None
user comments for change of rvst quarters

trading_scenario_hypo_mode_comment: str = None
comment for hypo mode choice

trading_scenario_n_ind_comment: str = None
comment for n ind choice

wal_monitor_choice: float = None
user choice WAL value for the monitor test only (seperate from other WAL/tenor assumptions)

wal_monitor_comment: str = None
comment for wal monitor choice

warf_ind_port_choice: float = None
WA rating factor of the indicative portfolio

warf_ind_port_comment: str = None
WA rating factor of the indicative portfolio

was_high_choice: float = None
not used

was_ind_port_choice: float = None
WA spread of the indicative portfolio

was_ind_port_comment: str = None
WA spread of the indicative portfolio

was_low_choice: float = None
trading scenario user choice for spread tightening

was_low_comment: `str = None`
comment for WAS low choice

class InputStep4a(*repline_trading_pct_choice=None*, *rl_recoveries_trading=None*, *repline_trading_pct_choice_mode_3=None*, *scenarios=None*, *trading_scenario_is_on_choice=None*, *deal_matrix_dbrs=None*, *deal_matrix_dbrs_warr_rtg=None*, *deal_matrix_dbrs_cf_attributes=None*, *deal_matrix_dbrs_cf_levels=None*, *deal_matrix_dbrs_cf_values=None*, *wac=None*, *trading_scenario_is_on_comment=None*, *repline_trading_pct_mode3_comment=None*, *amort_tail_choice_matrix=None*, ***kwargs*)

Bases: `object`

Master data structure that represents the inputs to *model_clo_insight.app.step_4a_generate_cf_scenarios()*

amort_tail_choice_matrix: `float = None`
user choice for the amortization schedule lookup

cl_cov_lite: `float = None`
max cov-lite loans

Type concentration limitation

cl_second_lien: `float = None`
max second lien loans

Type concentration limitation

deal_matrix_dbrs:
List[*model_clo_insight.clo_external_data_contracts.DBRSMatrixInputs*] = None
instances of *DBRSMatrixInputs* that represent the transaction's collateral quality matrix if structured to DBRS's methodology (trading_scenario_mode = 3)

deal_matrix_dbrs_cf_attributes: `List[str] = None`
Cash Flow Dynamic Matrix attributes if available for MM deal (tranche Par or coverage Test)

deal_matrix_dbrs_cf_levels: `List[int] = None`
Cash Flow Dynamic Matrix attribute levels if available for MM deal (e.g. which tranche)

deal_matrix_dbrs_cf_values: `List[float] = None`
Cash Flow Dynamic Matrix data if available for MM deal

deal_matrix_dbrs_warr_rtg: `str = None`
for deals structured to DBRSM's methodology (trading_scenario_mode = 3), the rating stress level that the WA recovery rate test was sized to

repline_trading_pct_choice:
List[*model_clo_insight.clo_external_data_contracts.CFReplineTradingPct*] = None
repline percentages user choice for the trading scenario recommendation (modes 1 & 2)

repline_trading_pct_choice_mode_3: `List[float] = None`
repline percentages user choice for the trading scenario recommendation (mode 3)

repline_trading_pct_mode3_comment: `List[str] = None`
comment for mode 3 replines chosen

rl_recoveries_trading:
List[*model_clo_insight.clo_external_data_contracts.RatingVectorData*] = None
instances of *RatingVectorData* that represent the recovery rate assumptions for the trading scenario replines

scenarios: List[*model_clo_insight.clo_external_data_contracts.TradingScenarioRow*] = None
instances of *TradingScenarioRow* that represent the trading scenarios generated by step 3

trading_scenario_is_on_choice: List[*model_clo_insight.clo_external_data_contracts.TradingScenarioIsOnRow*] = None
instances of *TradingScenarioIsOnRow* that represent the user's choice if the scenario is active

trading_scenario_is_on_comment: List[str] = None
comment for trading scenario chosen

wac: float = None
weighted average coupon for generating scenario for fixed scenario

class InputStep4b(*hypo_diversity_mode=None*, *hypo_n_simple_ind=None*, *amort_tail_choice_matrix=None*, *rvst_per_choice_matrix=None*, *repline_trading_pct_choice=None*, *tranche_results=None*, *monitor_coef_bdr=None*, *monitor_coef_sdr=None*, *force_rtg_hi_choice=None*, *force_rtg_lo_choice=None*, *scenarios=None*, *trading_scenario_is_on_choice=None*, *starting_period_choice=None*, *wal_monitor_choice=None*, *starting_period_comment=None*, ***kwargs*)

Bases: object

Master data structure that represents the inputs to *model_clo_insight.app.step_4b_run_trading_scenarios()*

amort_tail_choice_matrix: float = None
user choice for the amortization schedule lookup

force_rtg_hi_choice: *model_clo_insight.clo_external_data_contracts.ForceRating* = None
instance of *ForceRating* that represents the high rating category to be fixed prior to solving for the rating mixture

force_rtg_lo_choice: *model_clo_insight.clo_external_data_contracts.ForceRating* = None
instance of *ForceRating* that represents the low rating category to be fixed prior to solving for the rating mixture

hypo_diversity_mode: int = None
determines how the industries are allocated in the hypo pool generator. 1 = multiple of obligor base conc limit, 2 = repeating sequence of N industries

hypo_n_simple_ind: int = None
for hypo_diversity_mode_2, the N industries to repeat

monitor_coef_bdr: List[float] = None
BDR coefficients for the CDO Monitor test. For new transactions, this is determined by analyzing comparable transactions

monitor_coef_sdr: List[float] = None
SDR coefficients for the CDO Monitor test. Typically this does not change deal to deal

reinvestment_periods_choice: int = None
user choice for number of periods in the cash flow modeling that covers the reinvestment period

repline_trading_pct_choice: List[*model_clo_insight.clo_external_data_contracts.CFReplineTradingPct*] = None
user choice for repline percentages recommended for the trading scenario recommendation

rvst_per_choice_matrix: int = None
user choice for forward start period in the simulation (e.g. reduce tenor in reinvestment period by this amount)

scenarios: List[*model_clo_insight.clo_external_data_contracts.TradingScenarioRow*] = None
instances of *TradingScenarioRow* that represent the trading scenarios generated from step 3

starting_period_choice: int = None
user choice for surveillance starting period in Proprietary Cashflow Engine for the trading scenarios

starting_period_comment: str = None
starting period comment

trading_scenario_is_on_choice:
List[*model_clo_insight.clo_external_data_contracts.TradingScenarioIsOnRow*] = None
instances of *TradingScenarioIsOnRow* that represent the user's choice if the scenario is active

tranche_results:
List[*model_clo_insight.clo_external_data_contracts.CLOTrancheOutputStep2*] = None
instances of *CLOTrancheOutputStep2* that represent the tranche data inputs from output_step_2

wal_monitor_choice: float = None
user choice WAL value for the monitor test only (seperate from other WAL/tenor assumptions)

class **InputStep5**(*tranche_results=None*, *deal_matrix_dbrs=None*, *scenarios=None*, *trading_scenario_is_on_choice=None*, *default_hurdle_matrix=None*, *cf_meta_data=None*, *bdr_matrix=None*, *cf_scenario_data=None*, *cf_scenario_lookup=None*, ***kwargs*)
Bases: object

bdr_matrix: List[List[float]] = None
matrix of break even default rates (BDRs) from Proprietary Cashflow Engine's dynamic runner

cf_meta_data: List[*model_clo_insight.clo_external_data_contracts.CFMatrixMetaData*] = None
instances of *CFMatrixMetaData* that represent the Proprietary Cashflow Engine cash flow scenario meta data

cf_scenario_data: List[List[float]] = None
values for cash flow dynamic runner

cf_scenario_lookup: List[int] = None
lookup that maps each Trading Scenario row to the unique CF scenario

deal_matrix_dbrs:
List[*model_clo_insight.clo_external_data_contracts.DBRSMatrixInputs*] = None
instances of *DBRSMatrixInputs* that represent the transaction's collateral quality matrix if structured to DBRSM's methodology (trading_scenario_mode = 3)

default_hurdle_matrix: List[List[float]] = None
matrix of cumulative default hurdle rates from the predictive model simulation

scenarios: List[*model_clo_insight.clo_external_data_contracts.TradingScenarioRow*] = None
instances of *TradingScenarioRow* that represent the trading scenarios generated from step 3

trading_scenario_is_on_choice:
List[*model_clo_insight.clo_external_data_contracts.TradingScenarioIsOnRow*] = None
instances of *TradingScenarioIsOnRow* that represent the user's choice if the scenario is active

tranche_results:
List[*model_clo_insight.clo_external_data_contracts.CLOTrancheOutputStep2*] = None
instances of *CLOTrancheOutputStep2* that represent the tranche data inputs (BDR, final maturity)

class JsonMetaData(*guid=None*, *time_stamp=None*, *deal_name=None*, *run_type=None*, *run_name=None*, *user_name=None*, *davinci_ind_guid=None*, *davinci_ind_run_name=None*, *davinci_scenarios_guid=None*, *davinci_scenarios_run_name=None*, *davinci_scenarios_guid_supplemental=None*, *davinci_scenarios_run_name_supplemental=None*, *user_notes=None*, *model_version=None*, ***kwargs*)
Bases: object

Contains the meta data to organize the json input/output files, and Proprietary Cashflow Engine run references

davinci_ind_guid: str = None
reference guid to the Proprietary Cashflow Engine run used for the indicative portfolio analysis

davinci_ind_run_name: str = None
reference run name to the Proprietary Cashflow Engine run used for the indicative portfolio analysis

davinci_scenarios_guid: str = None
reference guid to the Proprietary Cashflow Engine run used for the trading scenario matrix analysis

davinci_scenarios_guid_supplemental: List[str] = None
additional reference guids to the Proprietary Cashflow Engine run used for the trading scenario matrix analysis

davinci_scenarios_run_name: str = None
reference run name to the Proprietary Cashflow Engine run used for the trading scenario matrix analysis

davinci_scenarios_run_name_supplemental: List[str] = None
additional reference run names to the Proprietary Cashflow Engine run used for the trading scenario matrix analysis

deal_name: str = None
transaction name

guid: str = None
unique identifier

model_version: str = None
CLO Insight Model verion

run_name: str = None
run name

run_type: str = None

time_stamp: str = None
data and time stamp (set programatically)

user_name: str = None
the user's Windows domain name; set programatically

user_notes: str = None
free text for the user to jot down any notes on the run

class MethodologyAssumptions(*correl_matrix*, *recovery_lags_clo*, *n_cf_scens_per_tranche*)
Bases: object

Represents the methodology assumptions

correl_matrix: List[List[float]]
correlation assumptions (inter/intra industry & region). Regional correlation assumptions imply no diversification benefit.

n_cf_scens_per_tranche: int
number of cash flow scenarios per tranche

recovery_lags_clo: Dict[int, int]
recovery lags by country tier, used in repline construction for cash flow modeling

class MethodologyParameters(*correlation_matrix=None*, *recovery_lags=None*, ***kwargs*)
Bases: object

Represents the methodology parameters that can be overriden

correlation_matrix: List[List[float]] = None
correlation assumptions (inter/intra industry & region). Regional correlation assumptions imply no diversification benefit.

recovery_lags: List[int] = None
recovery lags by country tier, used in repline construction for cash flow modeling

class ModelParameters(*meth_assumptions*, *settings*, *recommended_vals*, *country_tier_tbl*, *recoveries_tbl*, *amort_tbl*, *amort_tbl_bullet*, *amort_tbl_eu*, *amort_tbl_half*, *trading_repline_tbl*, *agency_recoveries_tbl*, *scenario_shell_tbl*, *dbrsm_industry_tbl*, *performance_drift_table_dict*)
Bases: object

Master data structure that contains all of the parameters necessary to run each step of the model

agency_recoveries_tbl: Dict[str, Dict]
recovery rate assumptions used by other rating agency constraints in modes 1 & 2

amort_tbl: Dict[str, Dict]
table of amortization schedules by WAL covenant tail

amort_tbl_bullet: Dict[str, Dict]
bullet version of the amortization schedules. Used for testing.

amort_tbl_eu: Dict[str, Dict]
EU table of amortization schedules by WAL covenant tail

amort_tbl_half: Dict[str, Dict]
half speed of US table of amortization schedules by WAL covenant tail

build_dataframes()

country_tier_tbl: Dict[str, Dict]
lookup table for countries and their respective DBRSM tier

dbrsm_industry_tbl: Dict[str, Dict]
general shell for DBRSM industry

meth_assumptions:
model_clo_insight.clo_external_data_contracts.MethodologyAssumptions
methodology assumptions including correlation matrix, recovery lags

performance_drift_table_dict: Dict[str, Dict]
general shell for performance drift table

recommended_vals:
model_clo_insight.clo_external_data_contracts.TradingScenarioRecommendedValues
recommended values for trading scenarios like dscore high and low, monitor wal, max warf drift

recoveries_tbl: Dict[str, Dict]
recovery assumptions by seniority, DBRSM tier, and rating stress

scenario_shell_tbl: Dict[str, Dict]
general shell for trading scenarios

settings: *model_clo_insight.clo_external_data_contracts.ModelSettings*
model settings like custom pd curves, amort curve choice

trading_repline_tbl: Dict[str, Dict]
general shell for trading replines

class ModelPoolRow(*obligor=None*, *security=None*, *ind_code=None*, *numerical_rtg=None*, *par=None*, ***kwargs*)

Bases: object

Table row for the modeling pool (post-step 1, and hypo pools)

ind_code: int = None
DBRSM industry code

numerical_rtg: float = None
DBRSM rating in integer form

obligor: str = None
unique obligor identifier

par: float = None
par amount for this security

security: str = None
unique security identifier

class ModelPoolRowAssetSpecific(*obligor=None*, *security=None*, *ind_code=None*, *numerical_rtg=None*, *par=None*, *recovery_rate=None*, *tenor=None*, *custom_pd_curve=None*, ***kwargs*)

Bases: object

Asset specific input for the modeling pool (post-step 1, and hypo pools)

custom_pd_curve: str = None
asset specific assigned custom pd curve

ind_code: int = None
DBRSM industry code

numerical_rtg: float = None
DBRSM rating in integer form

obligor: str = None
unique obligor identifier

par: float = None
par amount for this security

recovery_rate: float = None
asset specific recovery rate

security: str = None
unique security identifier

tenor: float = None
asset specific tenor

class ModelSettings(*mc_seed=None*, *n_bins=None*, *n_trials=None*, *hypo_obligor_floor=None*, *hypo_industry_floor=None*, *monitor_solver_mode=None*, *monitor_cushion=None*, *monitor_rs_lower_bound=None*, *monitor_rs_upper_bound=None*, *ppy_is_off=None*, *amort_curve_choice=None*, *is_forward_start_AAA=None*, *is_stochastic_recoveries=None*, *is_asset_specific_rec=None*, *is_asset_specific_tenor=None*, *is_custom_pd=None*, *custom_pd_curves=None*, *cloam_setting=None*, *step4a_par_selection=None*, ***kwargs*)

Bases: object

Represents the model settings

amort_curve_choice: str = None
either US, EU, Bullet, or US Half speed

cloam_setting: bool = None
sets whether we are running step 2 as a cloam model

custom_pd_curves: Dict[float, List] = None
Custom PD curves

hypo_industry_floor: float = None
floor for the base industry concentration limitation (during solver)

hypo_obligor_floor: float = None
floor for the base obligor concentration limitation (during solver)

is_asset_specific_rec: bool = None
if True, each asset will use specific recovery rate

is_asset_specific_tenor: bool = None
if True, each asset will use specific tenor for recovery and pd and computations

is_custom_pd: bool = None
sets whether or not there are custom pd curves to begin with

is_forward_start_AAA: bool = None
if True, forward start will be used for AAA, otherwise starting period will be 0

is_stochastic_recoveries: bool = None
if True, assume stochastic recoveries (no current functional use)

mc_seed: int = None
seed for random number generator

monitor_cushion: float = None
BDR - SDR. Typically this is set to zero

monitor_rs_lower_bound: float = None
lower bound for the max DBRSM risk score constraint when running the monitor solver

monitor_rs_upper_bound: float = None
upper bound for the max DBRSM risk score constraint when running the monitor solver

monitor_solver_mode: str = None
Expected portfolio default rate (EPDR) or WA Rating Factor (SP_WARF)

n_bins: int = None
number of bins for the default distribution histogram

n_trials: int = None
number of trials for the simulation

ppy_is_off: bool = None
if True, turn off prepays (used for testing the old methodology)

step4a_par_selection: *str* = **None**
for step 4a, decide to use pool par or conc limit basis

class OutputStep1(*replines=None*, *rl_recoveries=None*, *rl_recoveries_trading=None*, *modeling_portfolio=None*, *tranche_data=None*, *was_high_recommend=None*, *was_low_recommend=None*, *dscore_high_recommend=None*, *dscore_low_recommend=None*, *AAA_was_recommend=None*, *repline_trading_pct_recommend=None*, *rec_rates_agency_recommend=None*, *force_rtg_hi_recommend=None*, *force_rtg_lo_recommend=None*, *amort_tail_recommend_indicative=None*, *rvst_per_recommend_indicative=None*, *amort_tail_recommend_matrix=None*, *rvst_per_recommend_matrix=None*, *prepay_start_recommend=None*, *wal_monitor_recommend=None*, *max_warf_drift_recommend=None*, *starting_period_recommend=None*, *pool_total_par=None*, *pool_perf_par=None*, *pool_def_par=None*, *dbrs_risk_score_ind_pool=None*, *warf_ind_port=None*, *WARF_mdy=None*, *dscore_ind_port=None*, *was_ind_port=None*, *wa_cpn=None*, *wa_life=None*, *wa_pd=None*, *total_tranche_draws=None*, *total_collat_px=None*, *davinci_prin_proceeds=None*, *strats_seniority=None*, *strats_cpn_type=None*, *strats_cov_lite=None*, *strats_country_tier=None*, *strats_country=None*, *strats_obligor=None*, *strats_industry=None*, *strats_dbrs_rating=None*, ***kwargs*)

Bases: object

Master data structure that represents the outputs from *model_clo_insight.app.step_1_parse_pool()*

AAA_was_recommend: List[float] = **None**
entire permitted WAS range as per the transaction documents. User chooses how granular to discretize

WARF_mdy: *float* = **None**
indicative portfolio - WARF using Moodys ratings only

amort_tail_recommend_indicative: *float* = **None**
recommended value for the amortization schedule lookup

amort_tail_recommend_matrix: *float* = **None**
recommended value for the amortization schedule lookup

davinci_prin_proceeds: *float* = **None**
if deal out of reinvestment period, use for Proprietary Cashflow Engine starting period section

dbrs_risk_score_ind_pool: *float* = **None**
indicative portfolio - WA DBRSM Risk Score

dscore_high_recommend: *int* = **None**
trading scenario recommendation for increase in diversity

dscore_ind_port: *float* = **None**
indicative portfolio - WA diversity score

dscore_low_recommend: *int* = **None**
trading scenario recommendation for decrease in diversity

force_rtg_hi_recommend: *model_clo_insight.clo_external_data_contracts.ForceRating* = **None**
instance of *ForceRating* that represents the high rating category to be fixed prior to solving for the rating mixture

force_rtg_lo_recommend: *model_clo_insight.clo_external_data_contracts.ForceRating* = **None**
instance of *ForceRating* that represents the low rating category to be fixed prior to solving for the rating mixture

max_warf_drift_recommend: float = None
recommended absolute max WARF increase in the trading scenarions (e.g. regardless of monitor/matrix, WARF cannot increase by more than this)

modeling_portfolio: List[*model_clo_insight.clo_external_data_contracts.ModelPoolRow*] = None
instances of *ModelPoolRow* that represent the modeling portfolio (e.g. ratings & industries mapped)

pool_def_par: float = None
indicative portfolio - total defaulted par

pool_perf_par: float = None
indicative portfolio - total performing par

pool_total_par: float = None
indicative portfolio - total par

prepay_start_recommend: int = None
recommended periods in the cash flow modeling to start prepayments

rec_rates_agency_recommend: List[*model_clo_insight.clo_external_data_contracts.AgencyRecovery*] = None
recommended values for generic non-DBRS rating agency recovery rate assumptions

repline_trading_pct_recommend: List[*model_clo_insight.clo_external_data_contracts.CFReplineTradingPct*] = None
repline percentages recommended for the trading scenario recommendation

replines: List[*model_clo_insight.clo_external_data_contracts.CFRepline*] = None
instances of *CFRepline* that represent the cash flow replines

rl_recoveries: List[*model_clo_insight.clo_external_data_contracts.RatingVectorData*] = None
instances of *RatingVectorData* that represent the recovery rate assumptions for the indicative portfolio replines

rl_recoveries_trading: List[*model_clo_insight.clo_external_data_contracts.RatingVectorData*] = None
instances of *RatingVectorData* that represent the recovery rate assumptions for the trading scenario replines

rvst_per_recommend_indicative: int = None
recommended number of periods in the cash flow modeling that covers the reinvestment period

rvst_per_recommend_matrix: int = None
recommended number of periods in the cash flow modeling that covers the reinvestment period

starting_period_recommend: int = None
recommended surveillance starting period in Proprietary Cashflow Engine for the trading scenarios

strats_country: List[*model_clo_insight.clo_external_data_contracts.PoolStratRow*] = None
Pool Stratifications by country

strats_country_tier: List[*model_clo_insight.clo_external_data_contracts.PoolStratRow*] = None
Pool Stratifications by country tier

strats_cov_lite: List[*model_clo_insight.clo_external_data_contracts.PoolStratRow*] = None
Pool Stratifications by cov lite

strats_cpn_type: List[*model_clo_insight.clo_external_data_contracts.PoolStratRow*] = None
Pool Stratifications by cpn type

strats_dbrs_rating: List[*model_clo_insight.clo_external_data_contracts.PoolStratRow*] = None
Pool stratifications by DBRSM rating (credit estimate vs rating)

strats_industry: List[*model_clo_insight.clo_external_data_contracts.PoolStratRow*] = None
Pool Stratifications by industry

strats_obligor: List[*model_clo_insight.clo_external_data_contracts.PoolStratRow*] = None
Pool Stratifications by obligor

strats_seniority: List[*model_clo_insight.clo_external_data_contracts.PoolStratRow*] = None
Pool Stratifications by seniority

total_collat_px: float = None
total amount of new collateral purchased under the modeling logic

total_tranche_draws: float = None
total amount of $ from undrawn tranches that the analyst assumed drawn

tranche_data: List[*model_clo_insight.clo_external_data_contracts.CLOTrancheOutputStep1*] = None
instances of *CLOTrancheOutputStep1* that represent the capital structure of the CLO

wa_cpn: float = None
indicative portfolio - WA fixed coupon

wa_life: float = None
indicative portfolio - WA life

wa_pd: float = None
indicative portfolio - WA default probability (lookup from the DBRSM IDT)

wal_monitor_recommend: float = None
recommended WAL value for the monitor test only (seperate from other WAL/tenor assumptions)

warf_ind_port: float = None
indicative portfolio - WARF using DBRSM-equivalent rating (e.g. avg of 2)

was_high_recommend: float = None
not used

was_ind_port: float = None
indicative portfolio - WA floating spread

was_low_recommend: float = None
trading scenario recommendation for spread tightening

class OutputStep2(*time_elapsed=None*, *pool_expected_EAD=None*, *implied_correlation=None*, *moments=None*, *default_hurdle_rates=None*, *cutpoint_tenors=None*, *tranche_results=None*, ***kwargs*)

Bases: object

Master data structure that represents the inputs to *model_clo_insight.app.step_2_run_indicative_pool()*

cutpoint_tenors: List[float] = None
tenor assumptions used for the IDT percentiling (expected tranche maturities)

default_hurdle_rates: List[float] = None
result of the predictice model - rating-based cumulative default rate hurdles

implied_correlation: float = None
implied correlation metric (Vasicek correlation that produces the same variance as the monte carlo)

moments: List[float] = None
statistical moments (mean, variance, skewness, kurtosis)

pool_expected_EAD: float = None
expected exposure-at-default (EAD)

time_elapsed: float = None
time in seconds to run the monte carlo simulation

tranche_results:
List[*model_clo_insight.clo_external_data_contracts.CLOTrancheOutputStep2*] = None
tranche summary result table

class OutputStep3(*scenarios=None*, *scenario_hypo_results=None*, ***kwargs*)
Bases: object

Master data structure that represents the outputs to *model_clo_insight.app.step_3_build_trading_scenarios()*

scenario_hypo_results:
List[*model_clo_insight.clo_external_data_contracts.TradingScenarioHypoResultRow*] = None
instances of *TradingScenarioHypoResultRow* that represent the results of the hypo pool generator for each *TradingScenarioRow*

scenarios: List[*model_clo_insight.clo_external_data_contracts.TradingScenarioRow*] = None
instances of *TradingScenarioRow* that represent the trading scenarios generated by step 3

class OutputStep4a(*cf_named_range=None*, *cf_scenario_lookup=None*, *cf_scenario_data=None*, *is_on_category=None*, ***kwargs*)
Bases: object

Master data structure that represents the outputs to *model_clo_insight.app.step_4a_generate_cf_scenarios()*

cf_named_range: List[*model_clo_insight.clo_external_data_contracts.CFNamedRangeRow*] = None
lookup key for cash flow modeling - Excel named ranges for dynamic runner

cf_scenario_data: List[List[float]] = None
values for cash flow dynamic runner

cf_scenario_lookup: List[int] = None
lookup that maps each Trading Scenario row to the unique CF scenario

is_on_category: List[str] = None
shows category of is on choices for each scenario

class OutputStep4b(*hypo_exception_results=None*, *hypo_exception_headers=None*, *default_hurdle_matrix=None*, *pooling_diagnostics=None*, ***kwargs*)
Bases: object

default_hurdle_matrix: List[List[float]] = None
matrix of cumulative default hurdle rates from the predictive model simulation

hypo_exception_headers: List[str] = None
headers of matrix for conc lim exceptions

hypo_exception_results:
List[*model_clo_insight.clo_external_data_contracts.HypoException*] = None
matrix of the realized concentration limitation exceptions from the hypo pool generator

pooling_diagnostics:
List[*model_clo_insight.clo_external_data_contracts.PoolingDiagnosticRow*] = None
instances of *PoolingDiagnosticRow* that represent the diagnostics from the simulation

class OutputStep5(*tranche_summary_results=None*, *detailed_diagnostics_bdr=None*, *detailed_diagnostics_hurdle=None*, *detailed_diagnostics_cushion=None*, *bdr_headers=None*, *hurdle_headers=None*, *cushion_headers=None*, ***kwargs*)
Bases: object

bdr_headers: List[str] = None
BDR headers

cushion_headers: List[str] = None
cushion headers

detailed_diagnostics_bdr:
List[*model_clo_insight.clo_external_data_contracts.DetailedDiagnosticRow*] = None
prints out bdr for each scenario

detailed_diagnostics_cushion:
List[*model_clo_insight.clo_external_data_contracts.DetailedDiagnosticRow*] = None
prints out cushions for each scenario

detailed_diagnostics_hurdle:
List[*model_clo_insight.clo_external_data_contracts.DetailedDiagnosticRow*] = None
prints out hurdle rates for each scenario

hurdle_headers: List[str] = None
hurdle headers

tranche_summary_results:
List[*model_clo_insight.clo_external_data_contracts.CLOTrancheOutputStep5*] = None
instances of *CLOTrancheOutputStep5* that represent the final tranche summary analysis across all the trading scenarios

class PoolStratRow(*Name=None*, *Count=None*, *Par=None*, *Percent=None*, ***kwargs*)
Bases: object

Table row that contains the pool strats outputs from step 1

Count: int = None
count of name type in indicative portfolio

Name: str = None
value being calculated, ie seniority, country etc

Par: float = None
total par of name type in indicative portfolio

Percent: float = None
percent of name type in indicative portfolio

class PoolingDiagnosticRow(*time_elapsed=None*, *max_risk_score=None*, *dscore=None*, *pool_expected_ead=None*, *implied_correl=None*, *moment_1=None*, *moment_2=None*, *moment_3=None*, *moment_4=None*, *AAA_wal=None*, *AA_wal=None*, *A_wal=None*, *BBB_wal=None*, *BB_wal=None*, *B_wal=None*, ***kwargs*)

Bases: object

Table row that contains the diagnostic info from the predictive model simulation

AAA_wal: float = None
effective weighted average life for AAA

AA_wal: float = None
effective weighted average life for AA

A_wal: float = None
effective weighted average life for A

BBB_wal: float = None
effective weighted average life for BBB

BB_wal: float = None
effective weighted average life for BB

B_wal: float = None
effective weighted average life for B

dscore: float = None
dscore from scenarios generated

implied_correl: float = None
implied correlation metric (Vasicek correlation that produces the same variance as the monte carlo)

max_risk_score: float = None
max risk score from scenarios generated

moment_1: float = None
EX - mean

moment_2: float = None
EX^2 - variance

moment_3: float = None
EX^3 - skewness

moment_4: float = None
EX^4 - kurtosis

pool_expected_ead: float = None
expected exposure-at-default (EAD)

time_elapsed: float = None
time (in seconds) that the simulation took

class RatingVectorData(*AAA=None*, *AAH=None*, *AA=None*, *AAL=None*, *AH=None*, *A=None*, *AL=None*, *BBBH=None*, *BBB=None*, *BBBL=None*, *BBH=None*, *BB=None*, *BBL=None*, *BH=None*, *B=None*, *BL=None*, *CCCH=None*, *CCC=None*, *CCCL=None*)

Bases: object

Used to store vectors of rating-based assumptions (e.g. recovery rates) or model outputs (e.g. default hurdle rates)

A: float = None
rating based assumption for A

AA: float = None
rating based assumption for AA

AAA: float = None
rating based assumption for AAA

AAH: float = None
rating based assumption for AAH

AAL: float = None
rating based assumption for AAL

AH: float = None
rating based assumption for AH

AL: float = None
rating based assumption for AL

B: float = None
rating based assumption for B

BB: float = None
rating based assumption for BB

BBB: float = None
rating based assumption for BBB

BBBH: float = None
rating based assumption for BBBH

BBBL: float = None
rating based assumption for BBBL

BBH: float = None
rating based assumption for BBH

BBL: float = None
rating based assumption for BBL

BH: float = None
rating based assumption for BH

BL: float = None
rating based assumption for BL

CCC: float = None
rating based assumption for CCC

CCCH: float = None
rating based assumption for CCCH

CCCL: float = None
rating based assumption for CCCL

class RecoveryLags(*tier1=None*, *tier2=None*, *tier3=None*, ***kwargs*)
Bases: object

list that shows overriden recovery lags

tier1: float = None
Recovery Lag for Tier 1

tier2: float = None
Recovery Lag for Tier 2

tier3: float = None
Recovery Lag for Tier 3

class TradingScenarioHypoResultRow(*credit_result=None*, *bdr=None*, *adj_bdr=None*, *sdr=None*, *monitor_cushion=None*, *risk_score=None*, *warf=None*, *credit_dispersion=None*, *dscore=None*, *ob_base=None*, *ind_base=None*, *n_assets=None*, *diversity_result=None*, ***kwargs*)
Bases: object

Table row for the results of the hypo pool solver

adj_bdr: float = None
CDO Monitor result adjusted for par loss/gain (as per the logic in the transaction documents)

bdr: float = None
CDO Monitor BDR result

credit_dispersion: float = None
weighted variance of the loan PDs

credit_result: int = None
True if a rating mixture was found that satisfies the constraints

diversity_result: str = None
detailed message of how the pool was solved

dscore: int = None
realized diversity score

ind_base: float = None

monitor_cushion: float = None
adj_bdr - sdr

n_assets: int = None
realized number of assets

ob_base: float = None
realized base obligor concentration limitation

risk_score: float = None
realized WA DBRSM risk score

sdr: float = None
CDO Monitor SDR result

warf: float = None
realized WARF

class TradingScenarioIsOnRow(*is_AAA=None*, *is_AA_BBB=None*, *is_BB_CCC=None*, ***kwargs*)
Bases: object

Table row for the users selection of which trading scenarios to turn on

is_AAA: bool = None
run the scenario for AAA rating stress

is_AA_BBB: bool = None
run the scenario for investment grade rating stress

is_BB_CCC: bool = None
run the scenario for sub-investment grade rating stress

class TradingScenarioRecommendedValues(*relative_was_drift*, *dscore_high*, *dscore_low*, *max_warf_drift*, *monitor_wal*, *trading_scenario_start_period*, *AAA_was_recommend*)

Bases: object

Contains the recommended values for the trading scenarios

AAA_was_recommend: List[float]
entire permitted WAS range as per the transaction documents. User chooses how granular to discretize

dscore_high: int
level of diversity score constraint in the increase divesrity scenario

dscore_low: int
level of diversity score constraint in the decrease divesrity scenario

max_warf_drift: float
maximum increase in WARF in the Max Risky scenarios

monitor_wal: float
weighted average life used in the Monitor calculations

relative_was_drift: float
relative % decline in the WAS (starting from current WAS)

trading_scenario_start_period: int
the period to assume the pro-forma trading scenarios start

class TradingScenarioRow(*scenario=None*, *spread_selection=None*, *credit_selection=None*, *pool_par=None*, *conc_limit_basis=None*, *was=None*, *dscore=None*, *seniority=None*, *warr=None*, *max_risk_score=None*, *amort_tail=None*, *is_AAA=None*, *is_AA_BBB=None*, *is_BB_CCC=None*, ***kwargs*)

Bases: object

Table row for a trading scenario

amort_tail: float = None
modeling amort tail years

conc_limit_basis: float = None
conc limit basis

credit_selection: str = None
as of close, max risky

Type WA credit risk

dscore: int = None
diversity score

is_AAA: int = None
run the scenario for AAA rating stress

is_AA_BBB: int = None
run the scenario for investment grade rating stress

is_BB_CCC: int = None
run the scenario for sub-investment grade rating stress

max_risk_score: float = None
WA credit risk

pool_par: float = None
pool par (varies for ramp-up CLOs)

scenario: int = None
id (1 to n)

seniority: int = None
seniority regime (1,2 or 3), used for modes 1 & 2

spread_selection: str = None
as of close, tighten, full range

Type WAS

warr: float = None
WA recovery rate, used for mode 3

was: float = None
WA spread (WAS)

4.2.3 model_clo_insight.clo_internal_objects module

This module serves as the location for all of the internal model object classes used in the computational engines.

Setup() method:

- used for assigning relationships based on other objects in the model

class ConcLimitException(*number*, *percentage*)
Bases: object

Concentration limitation exception

number: int
number of exceptions allowed at this level

percentage: float
concentration limitation level for a set number of exceptions

class CorpLoan(*ident*, *name*, *prop_id*, *par*, *tenor=None*, *recovery_rate=None*, *custom_pd_curve=None*)
Bases: object

Represents a Loan to a Corporation

amort_sched
amortization schedule for loan

contract_tenor
actual tenor of the Security, only used in the modeling if turned on in settings

Type float

ident
unique identifier for the loan

Type string

name
name identifier for loan

Type string

par
par balance of the corporate loan

Type float

parent_obligor
instance of the parent Obligor which secures this corporate loan

Type *Obligor*

pd
default probability, based on the PD curve of the obligor, tenor of security

pd_normed
pd converted to standard normal

prop_id
identifier for the parent Obligor

Type string

recovery_rate
asset specific recovery used for the Security, only used in the modeling if turned on in settings

Type float

reset()

set_amort(*raw_amort*, *settings*)
Compute amortization schedule for this Corp Loan

Parameters

- **raw_amort** – list[float]: array containing the remaining balance at the end of each year (optional)
- **settings** – takes in objects.ModelSettings

Returns amortization schedule

setup(*obligors*, *transaction*, *settings*)
Setup method for the CorpLoan class

CorpLoan needs access to the fields of its parent Obligor. This method also uses the Settings object to determine which tenor assumption to use. The security PD is normalized here so that the random normals in the simulation do not have to be converted to uniforms, which saves computation time

Parameters

- **obligors** (*list[clo_internal_objects.Obligor]*) – all Properties in the pool
- **transaction** – TransactionData: pool/deal data (custom PD curves)
- **settings** – ModelSettings: contains settings for tenor

Returns None

Notes

self.pd not used in the simulation, but is needed to compute pool metrics

tenor
float:

class HypoPoolRow(*row*, *ind_code*, *numerical_rtg*, *pct*, *total_par*, *pd_curve*)
Bases: object

Represents a single line item in a pool generated by the hypo pool generators

compute_pd(*tenor*, *pd_lookup*)

ind_code: int
DBRSM industry code

numerical_rtg: int
DBRSM rating in integer form

pct: float
percentage of total par for pool item

pd_curve: numpy.ndarray
cumulative pd curve (from IDT)

row: int
single line item of the pool

total_par: float
par amount for pool item

class MethodologyParamsDefaults(*IDT*, *rating_list*, *rtg_regime_map*, *correl*)
Bases: object

Contains the Methodology Parameters for Simulating Defaults in the Pooling Method

Notes

Typically these are stored in a `json` file inside the project

IDT
DBRS idealized default table

Type list of list

correl_diff_reg_diff_ind
depricated in methodology, but functionality available in model

correl_diff_reg_same_ind
depricated in methodology, but functionality available in model

correl_same_reg_diff_ind
asset correlation between different industries

Type float

correl_same_reg_same_ind
asset correlation within industry

Type float

rating_list
list of DBRS ratings

Type list

rtg_regime_map
each rating mapped to the whole rating

Type dict

class ModelSettings(*n_trials*, *seed*, *n_bins*, *is_stochastic_rec=False*, *is_asset_specific_rec=False*, *is_asset_specific_tenor=False*, *is_custom_pd=False*, *is_amort=False*)
Bases: object

Contains the User-Specified Model Settings for the Pooling Method

is_amort
if true, build asset-specific amortization schedules in terms of remaining balance

Type bool

is_asset_specific_rec
flag for using asset specific recovery assumptions

Type bool

is_asset_specific_tenor
flag for using asset specific tenors

Type bool

is_custom_pd
flag for using custom pd curves

Type bool

is_stochastic_rec
flag for using stochastic recoveries (not yet implemented)

Type bool

n_bins
number of bins to build the default/loss distribution

n_trials
number of trials for the simulation

Type int

seed
seed for the random number generator

Type int

class Obligor(*name*, *ind_code*, *numerical_rtg*, *pd_curve_ref=None*)
Bases: object

Represents the Obligor Securing a Corporate Loan

Data variables

_calc_total_par(*securities*)
Calculates total par across all securities belonging to this Obligor

Parameters **securities** – all securities belonging to obligor

Returns Total Par for all securities

custom_pd_curve
optional, used to lookup custom PD curve vs PD base on rating

Type string

ind_code
DBRSM industry code #testine float

Type int

name
unique identifier

Type string

numerical_rtg
DBRSM numerical rating (fractional implies split public ratings)

pd_curve
vector of PD curve over time

Type np.ndarray[float]

pd_curve_normed
normalized PD, used for time-to-default lookup

Type np.ndarray[float]

reset()
Not applicable for Obligor

Returns:

setup(*corp_loans*, *transaction*, *methodology*)
Setup method for the Obligor class

Obligors need to know the Corp Loans that belong to them in order to calculate total_par. This method also sets up the PD curve.

Parameters

- **corp_loans** (*list[clo_internal_objects.CorpLoan]*) – Corp Loans belonging to this Obligor (one Obligor can have many Corp Loans)
- **transaction** (clo_internal_objects.TransactionData) – pool/deal data (custom PD curves)
- **methodology** (clo_internal_objects.MethodologyParamsDefaults) – contains IDT for PD lookup from rating

Returns None

total_par
total par across all securities belonging to this Obligor

Type float

class PoolPosition(*row*, *ob_name*, *sec_name*, *ind_code*, *rtg*, *par*, *rec_rate*, *tenor*, *pd_curve*)
Bases: tuple

represents raw row of data for pooling inputs

_asdict()
Return a new dict which maps field names to their values.

_field_defaults = {}

_fields = ('row', 'ob_name', 'sec_name', 'ind_code', 'rtg', 'par', 'rec_rate', 'tenor', 'pd_curve')

_fields_defaults = {}

classmethod _make(*iterable*)
Make a new PoolPosition object from a sequence or iterable

_replace(***kwds*)
Return a new PoolPosition object replacing specified fields with new values

ind_code

ob_name

par

pd_curve

rec_rate

row

rtg

sec_name

tenor

class TransactionData(*pool_tenor*, *tranche_tenor_rolldown*, *tranche_tenor_level*, *tranche_tenor_rtg*, *pool_amort=None*, *custom_pd_dict=None*)
Bases: object

Contains the Transaction Level Data

Pool and deal-level information. This object also contains transaction-specific lookup tables and settings

applicable_ind_codes
unique list of industries across the pool

Type list

custom_pd_dict
dictionary to assign asset PD curves based on custom curve in the simulation method

pool_amort
pool amortization per rating regime

Type dict of lists

setup(*obligors*)
Setup method for the Transaction Data

Transaction needs to know the Property in the pool in order to determine which property types & MSA Groups to shock in the simulation.

Parameters **obligors** – list of Obligor: obligors in the pool

Returns:

tenors_agg
aggregate tenors across ratings stack

Type list of float

tranche_tenor_level
expected final maturity of each tranche/rating level

Type list of float

tranche_tenor_rtg
expected final maturity of each tranche/rating level

Type list of int

transaction_tenor
tenor of the pool, generally weighted average or covenant based

Type float

build_obs_and_loans_sp(*raw_pool*, *pool_wal*)
Used to build / debug hypo, so just assume some dummy parameters for setttings and transaction

4.2.4 model_clo_insight.utilities module

Module for utilities used across the code

clean_numerical_output(*x*)
Rounds numerical output for consistency between different clients

By rounding to 6 digits there should not be an issue when running pooling using different clients.

Parameters **x** – float

Returns: float

convert_rtg(*df*, *column_name*)
converts ratings for full column in Pandas dataframe to high/low format

Parameters

- **df** – Pandas Dataframe
- **column_name** – column we are converting ratings from

Returns corrected dataframe column of ratings

convert_single_rtg(*value*, *warr*)
converts single rating to high/low format

Parameters

- **value** – rating
- **warr** – if rating is used for WARR is deal matrices, we keep it as capitalized

Returns corrected rating

dataframe_dropna(*df*)
takes in dataframe and drops all NA values

Parameters **df** – Pandas Dataframe

Returns Pandas dataframe with no NA values

dataframe_to_list_of_dataclass(*df*, *my_dataclass*, *header=True*, *dropna=True*)
takes dataframe and converts it to a list of dataclasses

Parameters

- **df** – Pandas dataframe
- **my_dataclass** – dataclass format we want it to be in

Returns List of dataclass

json_encoder(*o*)
JSON encoder class.

Notes; Handling timestamp formatting and NaT from pandas.

Parameters **o** (*object*) – object to serialise

Returns encoded object

Return type object

listSerDe(*list_to_serialise*, *ignore_nan=True*)
Non Python native objects serialiser handler.

Notes; Numpy nan or datetime are not natively serialise by JSON (or not correctly). We do the SerDe ourselves to overcome this issue and stick wih native dictionary python as exit. Probably not the most efficient, but the faster option.

Parameters

- **list_to_serialise** (*Series, DataFrame*) – object to SerDe
- **ignore_nan** (*bool*) – ignore nan values

Returns SerDe Python dictionary

Return type dict

list_dropna(*ls*)
takes in list and drops all NA values

Parameters **ls** – list

Returns List with no NA values

list_of_dataclass_to_dataframe(*list_of_dataclass*)
takes list of dataclass and converts it to a Pandas dataframe

Parameters **dataclass** (*list of*) –

Returns Pandas dataframe

list_of_lists_dropna(*ls*)
takes in a list of lists and drops all NA values

Parameters **ls** – List of lists

Returns List of lists with no NA values

list_of_lists_to_dataclass(*df*, *my_dataclass*, *header=False*)
takes in list of lists and converts to dataclass

Parameters

- **df** – list of lists taken in
- **my_dataclass** – dataclass structure we are using

Returns dataclass object

Part III

Indices and tables

- genindex
- modindex

PYTHON MODULE INDEX

m

INDEX

Symbols

A

B

C

D

E

F

M

N

O

P

R

S

T

U

W