# Unit Testing

**Table of Content:**

1. Classes to be tested
2. Example Input & Output of each method to be tested
3. Example test cases for each method to be tested
4. List of all possible routes in the method
5. TODO Description

**I. Classes to Be tested:**

1. **Binning Machine**
   * Description: A class for performing binning based on bins settings & good bad definition
   * Public Methods:
     1. **perform\_binning\_on\_whole\_df(self, bins\_settings\_list)**
        + Description: A public method for performing binning for the whole dataframe based on bins\_settings, returns a binned\_df
     2. **perform\_binning\_on\_col(self, col\_df, bin\_method)**
        + Description: A public method for performing binning for a single column based on bins\_settings, returns a pd.Series
   * Private Methods:
     1. **\_\_perform\_eq\_width\_binning\_by\_width\_\_(self, col\_df, dtype, width)**
        + Description: A method for performing equal width binning with a specified width, returns a pd.Series
     2. **\_\_perform\_eq\_width\_binning\_by\_num\_bins\_\_(self, col\_df, dtype, num\_bins)**
        + Description: A method for performing equal width binning with a specified number of fixed-width bins, returns a pd.Series
     3. **\_\_perform\_eq\_freq\_binning\_by\_freq\_\_(self, col\_df, dtype, freq)**
        + Description: A method for performing equal frequency binning with a specified frequency, returns a pd.Series
     4. **\_\_perform\_eq\_freq\_binning\_by\_num\_bins\_\_(self, col\_df, dtype, num\_bins)**
        + Description: A method for performing equal frequency binning with a specified number of fixed-frequency bins, returns a pd.Series
     5. **\_\_perform\_custom\_binning\_\_(self, col\_df, dtype, bins\_settings)**
        + Description: A method for performing binning based on boundary points obtained from interactive binning, returns a pd.Series
2. **Stat Calculator**
   * Description: A class to calculate statistical values for displaying the mixed chart & statistical tables
   * Fields:
     1. **df**
        + Description: whole dataset
     2. **col\_bins\_settings**
        + Description: binning description of the column
     3. **good\_bad\_def**
        + Description: good and bad definitions for calculating good bad counts
   * Public Methods:
     1. **compute\_summary\_stat\_table(self)**
        + Description: Output a dataframe representing the summary statistics table of the column
   * Private Methods:
     1. **\_\_compute\_bin\_stats\_\_(self, bin\_df, bin\_name, total\_good, total\_bad)**
        + Description: Compute statistics for a bin
     2. **\_\_compute\_var\_stats\_\_(self, var\_summary\_df, total\_good, total\_bad)**
        + Description: Compute overall statistics of the variable
     3. **\_\_compute\_pct\_\_(self, value, total\_value)**
        + Description: Compute a percentage
     4. **\_\_compute\_odds\_\_(self, good, bad)**
        + Description: Compute the odds
     5. **\_\_compute\_info\_odds\_\_(self, good\_pct, bad\_pct)**
        + Description: Compute the info\_odds
     6. **\_\_compute\_woe\_\_(self, info\_odds)**
        + Description: Compute the WOE
     7. **\_\_compute\_mc\_\_(self, good\_pct, bad\_pct, woe)**
        + Description: Compute the MC
3. **Good Bad Counter**
   * Description: A class for counting the number of good and bad samples/population in the column
   * Public Methods:
     1. **get\_statistics(self, dframe, good\_bad\_def)**
        + Description: A method to get the number of sample bad, sample indeterminate, sample good, population good, and population bad
   * Private Methods:
     1. **\_\_count\_sample\_bad(self, dframe, bad\_defs)**
        + Description: A method to count the number of sample bad
     2. **\_\_count\_sample\_indeterminate(self, dframe, indeterminate\_defs)**
        + Description: A method to count the number of sample indeterminate
     3. **\_\_count\_sample\_good(self, dframe, sample\_bad\_count, sample\_indeterminate\_count)**
        + Description: A method to count the number of sample good
     4. **\_\_get\_population\_good(self, sample\_good\_count, good\_weight)**
        + Description: A method to count the number of population good
     5. **\_\_get\_population\_bad(self, sample\_bad\_count, bad\_weight)**
        + Description: A method to count the number of population bad
4. **Good Bad Def Validator**
   * Description: A class for validating user inputs for good bad definitions
   * Public Methods:
     1. **validate\_if\_numerical\_def\_overlapped(self, bad\_numeric\_list, indeterminate\_numeric\_list)**
        + Description: A method to validate if numerical definitions for bad/indeterminate has overlapped
     2. **validate\_if\_categorical\_def\_overlapped(self, bad\_categoric\_list, indeterminate\_categoric\_list)**
        + Description: A method to validate if categorical definitions for bad/indeterminate has overlapped
     3. **validate\_numerical\_bounds(self, numeric\_info\_list)**
        + Description: A method to validate if all numerical definition range have upper bound > lower bound, if not, returns false
5. **Good Bad Def Decoder**
   * Description: A class for obtaining user inputs from the section UI info
   * Public Methods:
     1. **get\_numeric\_def\_list\_from\_section(self, numeric\_info\_list)**
        + Description: A method to translate section UI info to a list of numerical definition
     2. **get\_categorical\_def\_list\_from\_section(self, categoric\_info\_list)**
        + Description: A method to translate section UI info to a list of categorical definition