EMeRGE downloads 0

EMeRGE (Emerging technologies Management and Risk evaluation on distribution Grids Evolution) is a collection of mini-tools to help users develop openDSS feeder model from GIS (.shp) file and perform risk analysis at various PV scenarios and visulize results in an interactive dashboard made using Dash.

Releases release v1.5

Installation

Run the following command to install:

```
pip install EMeRGE
```

Usage

`csvextractor` package (converts .shp files into CSV files)

1. Requirements
    1. All .shp files stored in a folder
    2. QGIS, an open source tool (3.8 version) installed
    3. Open QGIS python shell, copy and paste the code from csvextractor/converter.py
    4. Pass input_path and output_path to `GIS2CSV` class instance

`csvconverter` package (converts CSVs exported from `csvextractor` into standard formats)

2. Requirements
    1. Store all extracted CSV files from QGIS in a folder name 'GISCSVs'
    2. Store extra CSV files containing load profile data, solar profile data, linecode and wiredata (sample files available in github) in a folder names 'ExtraCSVs'

Creating a project skeleton for CSV formatting

```
     from csvconverter.formatter import Convert
    instance = Convert()
    instance.create_skeleton(<project_path>)
```

Converting CSVs into a standard format

Using config.json file

```
     from csvconverter.formatter import Convert
    instance = Convert('config.json')
```

Using python dict

```
     from csvconverter.formatter import Convert
    intance = Convert({'project_path':r'.\Project_formatter','feeder_name':'Test'})
```

`dssgenerator` package (generates dss files from CSV files)

3. Requirements:
    1. Standard CSV files generated using `csvconverter` stored in a folder name same as feeder name
    2. Extra CSV files containing load profile, voltage profile and solar profile data in a folder named 'ExtraCSVs'

Creating a project skeleon for generating DSS files

```
     from dssgenerator.generator import CSV2DSS
    intance = CSV2DSS()
    instance.create_skeleton(<project_path>,<feeder_name>)
```

Converting CSV files into DSS files

Using config.json file

```
 from dssgenerator.generator import CSV2DSS
instance = CSV2DSS('config.json')
```

Using using python dict

```
 from dssgenerator.generator import CSV2DSS
instance = CSV2DSS({'project_path':".",'feeder_name':'test'})
```

`dssmetrics` package (Compute metrics)

Exporting metrics for a single scenario

4. Requirements:
    1. OpenDSS files of distribution system
    2. Three pickle files (each file should contain python dictionary with element name as key and array of downward customer name as value) for 'Tranformer', 'Line' and 'Node'. Use pickle files generated by module CSV2DSS.
    3. Temperature data in the same resolution as simulation time period. (Optional)
    4. Transformer life parameters (Optional)
    5. Data from 4.2, 4.3 and 4.4 can be stored in a single folder and path can be specified in `config.json`

using config.json file

```
 from dssmetrics.opendss import OpenDSS
instance = OpenDSS('config.json')
instance.qsts_powerflow()
```

How to get `config.json` file for computing metrics for a single scenario

```
 from dssmetrics.opendss import OpenDSS
instance = OpenDSS()
```

Exporting metrics for a multiple PV scenarios

5. Requirements
    1. For each scenario create a folder inside `DSSScenarios` and copy dss files inside that folder
    2. Copy pickle files, temperature data (optional) and transformer life parameters (optional) inside `ExtraData` folder
    3. Inside `Category` folder create a new folder, copy and paste the `config.json` inside a new folder. Note before running code you have to do few edits on `config.json`

Create a project skeleton for computing metrics for multiple PV Scenarios

```
 from dssmetrics.main import MultipleOpenDSS
a = MultipleOpenDSS()
a.create_skeleton(<project_path>,<project_name>)
```

Computing metrics for multiple scenarios

using config.json file

```
 from dssmetrics.main import MultipleOpenDSS
instance = MultipleOpenDSS()
instance.simulate('config.json')
```

`dssdashboard` package (Visuallize data in dashboard)

6. Requirements
    1. `Coordinates` folder should contain coordinate csv files for line,node,customers and transformers

2. `PVMetrics` folder should contain results after computing metrics for various PV scenarios using `dssmetrics` package (without volt/var)
3. `AdvancedPVMetrics` folder should contain results after computing metrics for various PV scenarios using `dssmetrics` package (with volt/var)
4. `Profile` folder should contain profile for all consumer classes along with solar profil
5. `PVConnection` folder - should have two folders : 'Base' containing all dss files and 'ExtraData' containing pickle files and csv files (requirement same as dssmetric package)

Creating a project skeleton for creating dashboard

```
     from dssdashboard.dashboard import AppServer
    instance = AppServer()
    instance.create_skeleton(<project_path>,<feeder_name>)
```

Launching a dashboard

using config.json file

```
     from dssdashboard.dashboard import AppServer
    instance = AppServer('config.json')
    instance.launch(port=8060)
```

JSON format of config.json file (`csvconverter` package)

```
{
    "project_path" : "",
    "feeder_name" : "",
    "log_settings": {
        "save_in_file": false,
        "log_folder": "",
        "log_filename":"",
        "clear_old_log_file": true
        },
    "MVA_to_KVA_conversion_for_PT"  : "yes",
    "force_lt_be_three_phase" : "yes",
    "PTrow" :  0,
    "three_phase" : "RYB",
    "single_phase" : ["R","Y","B"],
    "height_of_top_conductor_ht" : 9,
    "height_of_top_conductor_lt" : 8,
    "ht_spacing" : 1,
    "lt_spacing" : 0.47,
    "geomtry_units" : "m",
    "Service_wire_single_phase" : {
        "conductor_spacing" : 0.47,
        "num_of_cond" : 2,
        "num_of_phases" : 1,
        "height_of_top_conductor": 8,
        "phase_conductor":"AAAC_4.0",
        "neutral_conductor" : "AAAC_4.0",
        "units": "m",
        "spacing": "vertical"
    },
    "Service_wire_three_phase": {
        "conductor_spacing" : 0.47,
        "num_of_cond" : 4,
        "num_of_phases" : 3,
        "height_of_top_conductor": 8,
        "phase_conductor": "AAAC_4.0",
        "neutral_conductor" : "AAAC_4.0",
        "units": "m",
        "spacing": "vertical"
    },
    "ht_three_phase" : {
        "conductor_spacing" : 1,
        "num_of_cond" : 3,
        "num_of_phases" : 3,
        "height_of_top_conductor": 9,
        "phase_conductor": "RABBIT_7/3.35",
        "neutral_conductor" : "RABBIT_7/3.35",
        "units": "m",
        "spacing":"vertical"
    },
    "Consumer_kv": {
        "ht_consumer_ll" : 11.0,
        "ht_consumer_phase" : 6.6,
        "lt_consumer_ll" : 0.44,
        "lt_consumer_phase" : 0.23
    },
    "load_type": {
        "lt_consumer" : "wye",
        "ht_consumer" : "delta"
    },
    "ht_line": {
        "node_file_name" : "Asset_HT_Line_node.csv",
        "attribute_file_name" : "Asset_HT_Line_attribute.csv"
    },
    "ht_cable": {
        "node_file_name" : "Asset_HT_Line_Cable_node.csv",
        "attribute_file_name" : "Asset_HT_Line_Cable_attribute.csv"
    },
    "lt_line":{
        "node_file_name" : "Asset_LT_Line_node.csv",
        "attribute_file_name" : "Asset_LT_Line_attribute.csv"
    },
    "lt_cable":{
        "node_file_name" : "Asset_LT_Line_Cable_node.csv",
        "attribute_file_name" : "Asset_LT_Line_Cable_attribute.csv"
    },
    "line_column_mapper": {
        "length" : ["SHAPE_Leng"],
        "phase" :  ["force","RYB"],
        "four_conductor_system" : ["3Ph Five wire system","3Ph Four wire system"],
        "three_conductor_system" : ["3Ph Three wire system"],
        "two_conductor_system" : ["1Ph Three wire system","1Ph Two wire system","2Ph Three wire system"],
        "phase_system" : ["HTL_PWS","HTLC_PWS","LTL_PWS","LTLC_PWS"],
        "csize" : ["HTL_CSIZE_","HTLC_CBL_S","LTL_CSIZE","LTLC_CBL_S"],
        "cname" : ["HTL_CNAME","HTLC_CBL_T","LTL_CNAME","LTLC_CBL_T"],
        "nsize" : ["LTL_N_CSIZ"],
        "nname" : ["LTL_N_CNAM"],
        "units" : ["force","m"],
        "spacing" : ["force","vertical"]
    },
    "distribution_transformer":{"file_name" : "Asset_Distribution_Transformer.csv"},
    "power_transformer":{"file_name" : "Asset_Power_Transformer.csv"},
    "transformer_column_mapper": {
        "ID"  : ["DT_ID","PTR_ID"],
        "KVA_cap" : ["DT_CC_KVA","PTR_CAP_MV"],
        "HV_KV" : ["DT_HVSV_KV","PTR_PRY_VO"],
        "LV_KV" : ["DT_LVSV_KV","PTR_SEC_VO"],
        "maxtap" : ["force","1.1"],
        "mintap" : ["force","0.9"],
        "tap" : ["force","1.0"],
        "numtaps" : ["force","10"],
        "prim_con" : ["force","delta"],
        "sec_con" : ["force","wye"],
        "vector_group" : ["force","Dyn11"],
        "%resistance" : ["force","0.75"],
        "%reactance" : ["force","7.5"],
        "%noloadloss" : ["force","0"],
        "phase" : ["force","RYB"],
        "x" : ["x"],
        "y" : ["y"]
    },
    "lt_consumer": {"file_name" : "Consumer_LT.csv"},
    "ht_consumer":{"file_name" : "Consumer_HT.csv"},
    "consumer_column_mapper": {
        "pf" : ["LTC_PF","HTC_PF"],
        "tariff_type" : ["LTC_TCODE","HTC_TCODE"],
        "phase": ["LTC_PHASE","HTC_PHASE"],
        "Sanctioned_load" : ["LTC_SLOAD_","HTC_SDEMAN"],
        "x" : ["x"],
        "y" : ["y"],
        "PeakMWload" :  1.2,
        "estimate_consumer_peakkw" : "yes"
    },
    "consumer_class_by_tariff":{
        "residential" : ["LT Tariff IA","LT Tariff I B","LT Tariff VI"],
        "commercial" : ["LT Tariff II-A","LT Tariff II-B(1)","LT Tariff II-C", "LT Tariff V","LT Tariff II-B(2"],
        "industrial" : ["LT Tariff III-A (1)", "LT Tariff III-B","TARIFF III"],
        "agricultural" : ["LT Tariff IV"]
    },
    "peak_contribution": {
        "residential" : 0.867,
        "commercial" : 0.105,
        "industrial" : 0.017,
        "agricultural" : 0.011
    },
    "tec_per_kw_by_consumer_type":{
        "residential" : 5937.831,
        "commercial" : 6168.84,
        "industrial" : 6206.385,
        "agricultural" : 6102.5
    }
}
```

Arguments

7. Definitions
    1. `project_path` - str, path to a project folder
    2. `feeder_name` - str, CSVs will be stored in this folder
    3. `log_settings` - dict, settings for logging
        1. `save_in_file` - bool, set to true if you want to save log info in a file
        2. `log_folder` - str, folder path for saving log file
        3. `log_filename` - str, log file name, must end with '.log'
        4. `clear_old_log_file` - bool, set to true if you want to clear old log file
    4. `MVA_to_KVA_conversion_for_PT` - str, set to "yes" if conversion is necessary
    5. `force_lt_be_three_phase` - str, set to 'yes' if you want to force lt lines to be three phase
    6. `PTrow` - int, in case of multiple PTs define which row number in CSV file
    7. `three_phase` - str, e.g. "RYB"
    8. `single_phase` - list, e.g. ['R', 'Y', 'B']
    9. `height_of_top_conductor_ht` - float
    10. `height_of_top_conductor_lt` - float
    11. `ht_spacing` - float
    12. `lt_spacing` - float
    13. `geomtry_units` - str, e.g. "m"
    14. `Service_wire_single_phase` - dict, service wire info
        1. `conductor_spacing` - float,
        2. `num_of_cond` - int
        3. `num_of_phases` - int
        4. `height_of_top_conductor` - int
        5. `phase_conductor` - str
        6. `neutral_conductor` - str 7 `units` - str
        7. `spacing` - str, can be only 'vertical' or 'horizontal'
    15. `Service_wire_three_phase` - dict, service wire info (three phase)
        1. `conductor_spacing` - float
        2. `num_of_cond` - int
        3. `num_of_phases` - int
        4. `height_of_top_conductor` - float
        5. `phase_conductor` - str
        6. `neutral_conductor` - str
        7. `units` - str, e.g. 'm'

    8. `spacing` - str, can only be 'vertical' or 'horizontal'

16. `ht_three_phase` - dict, ht conductor info
    1. `conductor_spacing` - float
    2. `num_of_cond` - int
    3. `num_of_phases` - int
    4. `height_of_top_conductor` - float
    5. `phase_conductor` - str
    6. `neutral_conductor` - str
    7. `units` - str
    8. `spacing` - str
17. `Consumer_kv` - dict
    1. `ht_consumer_ll` - float
    2. `ht_consumer_phase` - float
    3. `lt_consumer_ll` - float
    4. `lt_consumer_phase` - float
18. `load_type` - dict
    1. `lt_consumer` - str, can be either 'wye' or 'delta'
    2. `ht_consumer` - tr, can be either 'wye' or 'delta'
19. `ht_line` - dict
    1. `node_file_name` - str
    2. `attribute_file_name` - str
20. `ht_cable` - dict
    1. `node_file_name` - str
    2. `attribute_file_name` - str
21. `lt_line` - dict
    1. `node_file_name` - str
    2. `attribute_file_name` - str
22. `lt_cable` - dict
    1. `node_file_name` - str
    2. `attribute_file_name` - str
23. `line_column_mapper` - dict
    1. `length` - list
    2. `phase` - list
    3. `four_conductor_system` - list
    4. `three_conductor_system` - list
    5. `two_conductor_system` - list
    6. `phase_system` - list
    7. `csize` - list
    8. `cname` - list
    9. `nsize` - list
    10. `nname` - list
    11. `units` - list
    12. `spacing` - list
24. `distribution_transformer` - dict
    1. `file_name` - str
25. `power_transformer` - dict
    1. `file_name` - str
26. `transformer_column_mapper` - dict
    1. `ID` - list
    2. `KVA_cap` - list
    3. `HV_KV` - list
    4. `LV_KV` - list
    5. `maxtap` - list
    6. `mintap` - list
    7. `tap` - list
    8. `numtaps` - list
    9. `prim_con` - list
    10. `sec_con` - list
    11. `vector_group` - list
    12. `%resistance` - list
    13. `%reactance` - list
    14. `%noloadloss` - list
    15. `phase` - list
    16. `x` - list
    17. `y` - list
27. `lt_consumer` - dict
    1. `file_name` - str
28. `ht_consumer` - dict
    1. `file_name` - str
29. `consumer_column_mapper` - dict
    1. `pf` - list
    2. `tariff_type` - list
    3. `phase` - list
    4. `Sanctioned_load` - list
    5. `x` - list
    6. `y` - list

    7. `PeakMWload` - float
    8. `estimate_consumer_peakkw` - str, either 'yes' or 'no'
30. `consumer_class_by_tariff` - dict
    1. `residential` - list
    2. `commercial` - list
    3. `industrial` - list
    4. `agricultural` - list
31. `peak_contribution` - dict
    1. `residential` - float,
    2. `commercial` - float,
    3. `industrial` - float,
    4. `agricultural` - float
32. `tec_per_kw_by_consumer_type` -{
    1. `residential` - float,
    2. `commercial` - float,
    3. `industrial` - float,
    4. `agricultural` - float

JSON format of configuration file for dss conversion (`dssgenerator` package)

```
{
    "project_path" : "",
    "feeder_name" : "",
    "log_settings": {
        "save_in_file": false,
        "log_folder": "",
        "log_filename":"",
        "clear_old_log_file": true
        },
    "PV_customers_step" : 10,
    "PV_capacity_step" : 1,
    "number_of_monte_carlo_run" : 1,
    "export_pickle_for_risk_analysis" : "yes",
    "time_series_pf" : "yes",
    "num_of_data_points" : 35040,
    "minute-interval" : 15,
    "time_series_voltage_profile" : "yes",
    "voltage_csv_name" : "voltagemult.csv",
    "sourcebasekv" : 33.0,
    "sourcebasefreq" : 50,
    "sourcepu" : 1.0,
    "sourcezeroseq_impedance" : [0.001,0.001],
    "sourceposseq_impedance" : [0.001,0.001],
    "source_num_of_phase" : 3,
    "include_PV" : "yes",
    "PV_volt_label" : [0.44,0.23],
    "annual_PV_capacity_factor" : 0.25,
    "inverter_oversize_factor" : 0.9,
    "max_pu_irradiance" : 0.98,
    "no_reactive_support_from_PV" : "yes",
    "PV_cutin" : 0.05,
    "PV_cutout" : 0.05,
    "solar_csvname" : "solarmult.csv",
    "three_phase" : "RYB",
    "single_phase" : ["R","Y","B"],
    "random_phase_allocation" : "yes",
    "multi_threephase_for_lt" : "yes",
    "num_of_parallel_three_phase" : 3,
    "servicewire_phase_conductor_type" : "AAAC",
    "servicewire_phase_conductor_size" : "4.0",
    "phase_conductor_type_ht_consumer" : "RABBIT",
    "phase_conductor_size_ht_consumer" : "7/3.35",
    "service_wire_spacing" : "vertical",
    "ht_consumer_conductor_spacing" : "vertical",
    "units_for_coordinate" : "m",
    "service_wire_num_of_cond" : {
        "single_phase" : 2,
        "three_phase" : 4
    },
    "ht_consumer_conductor_num_of_cond": {"three_phase" : 3},
    "phase2num" :{"R" : 1,"Y" : 2,"B" : 3}
}
```

Arguments

8. Definitions
    1. `project_path` - str, folder path containing data
    2. `feeder_name` - str, feeder name
    3. `log_settings` - dict, settings for logging
        1. `save_in_file` - bool, set to true if want to save in file
        2. `log_folder` - str, folder where you want to save log file
        3. `log_filename` - str, log file name
        4. `clear_old_log_file` - bool, set to true if you want to clear the ol log file
    4. `PV_customers_step` - int, number of scenarios with unique percentage customers 5 `PV_capacity_step` - int, number of scenarios with unique percentage pv capacity
    5. `number_of_monte_carlo_run` - int, must be greater than or equlal to 1
    6. `export_pickle_for_risk_analysis` - str, either "yes" or "no"

7. `time_series_pf` - str, "yes" or "no", specifying no will generate dss files for snapshot powerflow only
8. `num_of_data_points` - int, number of data-point in time series
9. `minute-interval` - int, simulation time step
10. `time_series_voltage_profile` - str, "yes" or "no" - whether to include voltage profile or not
11. `voltage_csv_name` - str, csv file name of voltage profile
12. `sourcebasekv` - float, voltage in kV
13. `sourcebasefreq` - int, can be either 50 or 60
14. `sourcepu` - float, pu voltage of swing bus
15. `sourcezeroseq_impedance` - list, e.g. [0.001, 0.001]
16. `sourceposseq_impedance` - list e.g. [0.001, 0.001],
17. `source_num_of_phase` - int, number of phases of swing bus
18. `include_PV` - str, either "yes" or "no" include PV or not
19. `PV_volt_label` - list, voltage level at which PV is to be connected [0.44, 0.23]
20. `annual_PV_capacity_factor` - float, PV annual capacity factor
21. `inverter_oversize_factor` - float, inverter size over PV size
22. `max_pu_irradiance` - float, maximum pu irradiance
23. `no_reactive_support_from_PV` - str, either "yes" or "no"
24. `PV_cutin` - float, pv cutin as defined in opendss
25. `PV_cutout` - float, pv cutout as defined in opendss
26. `solar_csvname` - str, csv filename of solar profile
27. `three_phase` - str, e.g. "RYB"
28. `single_phase` - list, e.g. ['R', 'Y', 'B']
29. `random_phase_allocation` - str, either "yes" or "no" random phase allocation for customers
30. `multi_threephase_for_lt` - str, either "yes" or "no"
31. `num_of_parallel_three_phase` - int, number of parallel three phases
32. `servicewire_phase_conductor_type` - str, name of conductor to be used as service conductor e.g. "AAAC"
33. `servicewire_phase_conductor_size` - str
34. `phase_conductor_type_ht_consumer` - str
35. `phase_conductor_size_ht_consumer` - str
36. `service_wire_spacing` - str, either 'vertical' or 'horizontal'
37. `ht_consumer_conductor_spacing` - str, either 'vertical' or 'horizontal'
38. `units_for_coordinate` - str, e.g. "m"
39. `service_wire_num_of_cond` - dict
    1. `single_phase` - int, number of conductors
    2. `three_phase` - int, number of conductors
40. `ht_consumer_conductor_num_of_cond` - dict
    1. `three_phase` - int, number of conductors
41. `phase2num` - dict
    1. `R` - int e.g 1
    2. `Y` - int e.g 2
    3. `B` - int e.g 3

JSON format (Exporting metric for a multiple scenarios - `dssmetrics` package)

```
{
    "project_path": "",
    "active_project":"",
    "active_scenario": "",
    "dss_filename":"",
    "start_time":"2018-1-1 0:0:0",
    "end_time":"2018-1-2 23:30:0",
    "simulation_time_step (minute)": 60,
    "parallel_simulation":true,
    "parallel_process": 2,
    "frequency": 50,
    "upper_voltage": 1.1,
    "lower_voltage":0.9,
    "record_every": 4,
    "export_voltages": false,
    "export_lineloadings": false,
    "export_transloadings":false,
    "export_start_date": "2018-1-1 0:0:0",
    "export_end_date": "2018-1-2 0:0:0",
    "volt_var": {
                "enabled": false,
                "yarray": [0.44,0.44,0,0,-0.44,-0.44],
                "xarray": [0.7,0.90,0.95,1.05,1.10,1.3]
                },
    "log_settings": {
                    "save_in_file": true,
                    "log_filename":"system",
                    "clear_old_log_file": true
                    }
}
```

Arguments

9. Definitions
    1. `project_path` - str, path to a project folder
    2. `active_project` - str, name of feeder or project inside project folder
    3. `active_scenario` - str, name of scenario
    4. `dss_filename` - str, master dss filename
    5. `start_time` - str, simulation start time
    6. `end_time` - str, simulation end time
    7. `simulation_time_step (minute)` - int or float, simulation resolution
    8. `frequency` - int, can be either 50 or 60
    9. `upper_voltage` - float, must be >1.0 and represents overvoltage threshold
    10. `lower_voltage` - float, must be <1.0 and represents undervoltage threshold
    11. `record_every` - int, multiple of simulation time-step
    12. `export_voltages` : bool, set to true if you want to export voltges
    13. `export_lineloadings` : bool, set to true if you want to export line loadings
    14. `export_transloadings` : bool, set to true if toy want to export transformer loadings
    15. `export_start_date` : str, start date for export results ((must be in format "2018-1-1 0:0:0"))
    16. `export_end_date` : str, end date for export results (must be in format "2018-1-1 0:0:0"),
    17. `volt_var` : dict, volt/var parameters
        1. `enabled` : bool, set to true to enable volt var
        2. `yarray` : list, refer to opendss manual for volt-var yarray
        3. `xarray` : list, refer to opendss manual for volt-var xarray
    18. `log_setting` - dict, settings for logging
        1. `save_in_file` - bool, true if logs to be saved in file else false
        2. `log_folder` - str, folder path where you want to save log file
        3. `log_filename` - str, name of the log file
        4. `clear_old_log_file` - bool, clears old log file if exists

JSON format (Exporting metric for a single scenario - `dssmetrics` package)

```
{
    "dss_filepath": "",
    "dss_filename":"",
    "extra_data_path": "",
    "export_folder":"",
    "start_time":"2018-1-1 0:0:0",
    "end_time":"2018-6-1 0:0:0",
    "simulation_time_step (minute)": 15,
    "frequency": 50,
    "upper_voltage": 1.1,
    "lower_voltage":0.9,
    "record_every": 96,
    "export_voltages": false,
    "export_lineloadings": false,
    "export_transloadings":false,
    "export_start_date": "2018-1-1 0:0:0",
    "export_end_date": "2018-1-2 0:0:0",
    "volt_var": {
                "enabled": false,
                "yarray": [0.44,0.44,0,0,-0.44,-0.44],
                "xarray": [0.7,0.90,0.95,1.05,1.10,1.3]
                },
    "log_settings": {
                    "save_in_file": false,
                    "log_folder": "",
                    "log_filename":"",
                    "clear_old_log_file": true
                    }
}
```

Arguments

10. Definitions
    1. `dss_filepath` - str, folder path containing all .dss files
    2. `dss_filename` - str, master dss filename
    3. `extra_data_path` - str, folder path containing pickle files and .csv input files
    4. `start_time` - str, simulation start time
    5. `end_time` - str, simulation end time
    6. `simulation_time_step (minute)` - int or float, simulation resolution
    7. `frequency` - int, can be either 50 or 60
    8. `upper_voltage` - float, must be >1.0 and represents overvoltage threshold
    9. `lower_voltage` - float, must be <1.0 and represents undervoltage threshold
    10. `record_every` - int, multiple of simulation time-step
    11. `export_voltages` : bool, set to true if you want to export voltges
    12. `export_lineloadings` : bool, set to true if you want to export line loadings
    13. `export_transloadings` : bool, set to true if toy want to export transformer loadings
    14. `export_start_date` : str, start date for export results ((must be in format "2018-1-1 0:0:0"))
    15. `export_end_date` : str, end date for export results (must be in format "2018-1-1 0:0:0"),
    16. `volt_var` : dict, volt/var parameters 16.1. `enabled` : bool, set to true to enable volt var 16.2. `yarray` : list, refer to opendss manual for volt-var yarray 16.3. `xarray` : list, refer to opendss manual for volt-var xarray
    17. `log_setting` - dict, settings for logging 17.1. `save_in_file` - bool, true if logs to be saved in file else false 17.2. `log_folder` - str, folder path where you want to save log file 17.3. `log_filename` - str, name of the log file 17.4. `clear_old_log_file` - bool, clears old log file if exists

JSON format for creating dashboard (`dssdashboard` package)

```
 {
"project_path": "",
"active_project":"",
"time_step(min)":"",
"year": 2018,
"log_filename":"log.log",
"pv_connection": {
    "dss_filename": "",
    "simulation_time_step (minute)": 15,
    "frequency": 50,
    "upper_voltage": 1.1,
    "lower_voltage":0.9
}
}
```

Arguments

2. Definitions
    1. `project_path` - str, path to project folder
    2. `active_project` - str, name of project of folder inside project folder
    3. `time_step(min)` - str, time resolution in minute, same as simulatio time period
    4. `year` - int, set the year for load profile analysis
    5. `log_filename` - str, path to log file, if not log will be displayed on the screen
    6. `pv_connection` - dict, settings for running powerflow
        1. `dss_filename` : str, dss file name
        2. `simulation_time_step (minute)` : int, simulation resolution in minute,
        3. `frequency` : int, frequency can be only 50 or 60
        4. `upper_voltage` : float, upper voltage limit
        5. `lower_voltage` : float, lower voltage limit