**ENERGY MEASUREMENT SYSTEM**

The project aims to come up with a solution to well manage the energy usage inside an industrial environemnt where there is a high requirement for energy availability to keep the production lines runing especially for the critical industries that are essenitial for human life like medicines and food manufacutring. We already know that companies while trying to fill the market demande they can consume a lot of energy that sometimes don't need, and because companies don't have the mecanismes to control their needs in term of energy especially on blackout periods, therefore, this ends up with a down time for them when the power source are fluctuating or unavailable for whatever reason. The IEMS systems can play an important role in this area of application to well streamlining the management of the energy flow durant this critical situations.

**Project use case.**

This project will bring to the light to one possible solution of an IEMS using Machine Learnig techniques, this case is related to optimize and rationalize energy consumed with one specific department (Local refregeration [department responsible of generating cold for cooling systems and production need]) in a company blue print during the blackout periods when the principal power resources are down and alternative resources are very limitted.

**Setup.**

While the manual installation can walk you through around the various commands basics for each used packages in the project, chances are you might already know those commands and you don't want to bother yourself about taping every single command, so that's why we have provided the possibility to use make scripting, to make life easy for you. You find in the following the commands you will need to do this.

In case you have make installed in your system, for Linux based system it comes already installed in your system you don't need to install anything just skeep this part to {...}. For windows based systems there are multiple ways to get GNU make installed, like for example Cygwin, Nmake, Cmake..., however we recommand to go for chocolatey, we think it's the most straighforward way to install make for windows systems with less effort.

**chocolatey.**

First thing first, we will install chocolatey, make sure you are using the Powershell command as an admin,

Then run this command first :

Get-ExecutionPolicy

If it returns Restricted, then run :

Set-ExecutionPolicy AllSigned or Set-ExecutionPolicy Bypass -Scope Process

Now, to install chocolatey run the following command by coping it at once and past it in command line, then hit enter:

Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))

Please check chocolatey website for more guidance !

Make.

Now that you have chocolatey installed, we can install make by running the command :

choco install make --version=3.81

Once make installation is done , and assuming that you have downloaded the project files in your local machine it's very easy to workout everything.

To create the conda environemnt run the command :

- $ make create\_env

To setup the project run the command :

- $ make setup

Test units.

Testing code is important to garantee consistency and availabality of this project. Test units are devide in two main aspects: testing code syntax and style conferming to PEP8 standard, testing internal modules which are mainly functions used for implemeting the code.

environment testing.

To test Python environment run the command :

- $ make env\_test

Testing syntax & style.

To lint code scripts we are using flake8, just run the following command :

- $ make lint

Under the hood, make will go over the Makefile located in our directory which itself will chain to all coding resources in /src/. and /notebooks directories and will check the syntax and style of your code using flake8 to meet PEP8 standards.

**Model Dataset.**

The collected data used in this particular project use case is based on the author work experience and estimations.

**Setup.py**

|  |
| --- |
| import setuptools  from setuptools import find\_namespace\_packages, setup  def get\_install\_requirements():  with open("requirements.txt", "r", encoding="utf-8") as f:  reqs = [x.strip() for x in f.read().splitlines()]  reqs = [x for x in reqs if not x.startswith("#")]  return reqs  setup(  name='AIEMS',  version='0.1.4',  description='Ai-based Energy Managment System',  author='Ahmed Mabrouk',  author\_email='mabrook.amed@gmail.com',  license='CC0 1.0 Universal',  packages=setuptools.find\_namespace\_packages(include=['src', 'src.models','src.features', 'src.data'] ),  install\_requires= get\_install\_requirements()  ) |

**Test\_env.py**

|  |
| --- |
| import sys  REQUIRED\_PYTHON = "python3"  def main():  system\_major = sys.version\_info.major  if REQUIRED\_PYTHON == "python":  required\_major = 2  elif REQUIRED\_PYTHON == "python3":  required\_major = 3  else:  raise ValueError("Unrecognized python interpreter: {}".format(  REQUIRED\_PYTHON))  if system\_major != required\_major:  raise TypeError(  "This project requires Python {}. Found: Python {}".format(  required\_major, sys.version))  else:  print(">>> Development environment passes all tests!")  if \_\_name\_\_ == '\_\_main\_\_':  main() |