ietools: A Python Package for Industrial Engineering Applications

Ganapathy Natarajan
University of Wisconsin - Platteville
natarajang@uwplatt.edu

Abstract

This article introduces a Python package named `ietools`. This package is intended to be developed and used for industrial engineering and management applications. The current version has two main modules `EnggEcon` and `Decision`, and a supporting `utils` module. Other modules in development are discussed. A call for contributors is issued at the end.
Keywords: Python package, industrial engineering, management, decision

1 Introduction

According to the Python Software Foundation (PSF) Python is an easy to learn and often the first programming language learned by non-programmers. Python has been a major contributor in the open-source software landscape and has been integrated into the infrastructure of many major companies like Alphabet, Meta, and Amazon, to name a few.

Opportunity 1: With Python being integrated into business infrastructure increasingly, there is an opportunity for industrial engineering students and practitioners to use Python.

Python community of users rely heavily on contributions from other users in the form of Python packages. Packages take existing Python functionalities and provide easy to use off-the-shelf solutions for specialized applications. Some well known examples are `numpy, pandas, scikit-learn, and matplotlib`.

Opportunity 2: Industrial engineering related packages are not widely available - meaning a practitioner is forced to spend hours writing code to perform common IE tasks.

There are a few packages available, viz. `enginEcon, scikit-mcda`, etc. However, they are highly specific to the task they perform and are often not maintained and updated to fix bugs and issues.

Opportunities 1 and 2 put together provides the rationale for this work. Python is being increasingly integrated into organizations' IT infrastructure. IEs who are part of those orga-

nizations need to be able to use and manipulate the different packages that exist. However, the lack of an IE specific package makes this harder. So, an IE package aimed at students (so that they can learn before entering the workforce) and practitioners will help both these audiences immensely.

Introducing ietools - a Python package for commonly used industrial engineering and engineering management related tools. `ietools` package (current version 0.1) may be downloaded or viewed from these following places:

- PyPI (the Python package repository): https://pypi.org/project/ietools/
- GitHub: https://github.com/gana1984/ietools

2 Code Design and Quality

The code design is inspired from existing Python packages that are widely used, viz. `numpy`, `scikit-learn`.

2.1 Code Design

Code design follows object oriented design principles, without inheritance. Methods are designed to be consistently named and handled.

A single module may contain classes with similar or related functionality. The primary class has a few user accessible methods. In some modules, viz. Decision, the class has a `fit` method that can perform all required operations with a single command. Whenever a `fit` method is used, additional class attributes are updated such that the user may access these attributes - which provide the expected solution/answer.

Documentation and code formatting follows `pep-8` guidelines.

2.2 Code Quality

Code quality is maintained using unit tests. Only minimal testing has been performed in the 0.1 version, but testing is planned. Code issues may be created on GitHub and these issues will be addressed and improved in subsequent releases.

3 Current Functionalities

In this version, `EnggEcon` and `Decision` are two main modules that are implemented. An additional support module `utils` provides helper functions for the `EnggEcon` module.

3.1 `EnggEcon`

`EnggEcon` has the following classes:

- `equivalence` - basic methods to calculate equivalence values. Implements basic single cash flow, repeated cash flow, arithmetic gradient, and geometric gradient calculations.
- `compateAlt` - methods to compare cash flows. NPV, FPV, EUAW, and IRR may be calculated for cash flows. IRR is implemented using a Newton-Raphson root finding algorithm. Multiple IRR is not implemented in the current version and will be included in future versions.
- `BCRatio` - methods to calculate benefit-cost ratio. Different rates for the benefit cash flows and cost cash flows may be used.

All the engineering economy related calculations may be performed using interest rates represented either as a fraction or as a percentage. In order to use percentage numbers (and not fractional rates), the `fractional` parameter may be set to `False` when instantiating the class.

3.2 `Decision`

Decision making under uncertainty using different decision criteria is implemented in the `Decision` module. The following criteria are implemented: maxmax, maxmin, minmax regret, maximum likelihood, and expected value. Hurwicz criterion is not implemented in the current version, but will be incorporated in future versions.

Users provide a payoff matrix in the following format:

	State_1	State_2	...	State_n
Alternate_1	Payoff_11	Payoff_12	...	Payoff_1n
Alternate_2	Payoff_21	Payoff_22	...	Payoff_2n
⋮	⋮	⋮	⋮	⋮
Alternate_m	Payoff_m1	Payoff_m2	...	Payoff_mn
Probability	p(State_1)	p(State_2)	...	p(State_n)

The payoff matrix may be provided by the users in multiple ways - a path to a csv file, a `pandas` dataframe, a `dict`, or a `numpy` array. However, as noted above, the last line is expected to be the probability of states. In future implementations, users may be able to specify where the probabilities are located, as well a different orientation for the payoff table.

If the user wants to get decisions based on multiple criteria, then the `fit` method may be used to calculate the decision and payoff using all the criteria. `results_` parameter stores results for the multiple criteria as a `Dict` - dictionary object.

3.3 `utils`

There are two engineering economy related utility functions in the `utils` module.

The first method is `effect` that calculates the effective interest rate given an APR (`r`) and a compounding period (`m`).

The second method is `cfcommon`. This method is a helper method for NPV analysis, when the alternatives have different useful lives. The method takes a `Dict` input of different cash flows. The method calculates the LCM of useful life and automatically generates cash flows of equivalent useful lives. The original dictionary is modified to store the equivalent cash flows. LCM is used as the common time horizon in this implementation. Future versions will allow the user to specify a different criteria.

4 Examples of Current Functionalities

The following code snippet shows the use of `EnggEcon` in calculating a NPV and IRR for a given cash flow.

```
from ietools import EnggEcon as ee
cf = [-1000, 200, 300, 400, 500 ]
alt1 = ee.compareAlt(cf)
interest = 0.1
print("NPV: {:.2f}".format(alt1.npv(interest)))
print("IRR: {:.4f}".format(alt1.irr()))
```

The output of the previous code snippet, would be:

```
NPV: 71.78
IRR: 0.1283
```

Here is an example of using the `Decision` module. We first need the payoff table. The file `payoff.csv` has the following data:

	Competitor $6	Competitor $7	Competitor $8
Charge $5	125	175	225
Charge $6	200	300	400
Charge $7	225	375	525
Charge $8	200	400	600
Charge $9	125	375	625
Prob	0.35	0.25	0.4

Now that we have the payoff table, we can use our `Decision` module to compute the results using all of the available decision criteria.

```
from ietools.Decision import Decision
import pandas as pd
payoff = pd.read_csv('Payoff.csv')
result = Decision(payoff)
result.fit()
print(result.results_)
```

The above snippet of code would provide the following output.

```
{'maxmax': ('Charge $9', 625.0), 'maxmin': ('Charge $7', 225.0), 'regret':
```

('Charge $8', 25.0), 'maxlik': ('Charge $9', 625.0), 'ev': ('Charge $8', 625.0)}

The results_ dictionary stores results in the format of {criteria:(decision,payoff)}

5 Future Implementations

Table 1 shows future modules that are planned, underlying technologies used in those modules, and some features that are planned in those modules.

Table 1: Implementation Planned and Underlying Technologies

Module Name	Underlying Technolog(ies)y	Planned Features
Inventory	numpy, scipy	EOQ, News-vendor Model, (Q,R) models, ABC models
Aggregate	numpy, scipy, pulp, pandas	Aggregate planning LP, Chase Strategy, Level Strategy
Forecasting	scipy,stastmodel, matplotlib	Moving averages, exponential smoothing models, ARIMA, and growth curves
SQC	numpy, scipy, matplotlib, seaborn	Control limits, control charts, other quality metrics and models
Flow	numpy, scipy, pandas, matplotlib	Flow system analysis, Factory Physics

Other than these modules, other modules may be developed based on demand and input from users. The development process will follow Agile methodology with fast development times and improvements incorporated in subsequent iterations.

6 Intended Audience and Use

The intended audience for this package are educators, students, and IE practitioners. Students learning these modules will help them as they transition into being practitioners. Current practitioners are expected to use the package and contribute to its improvement based on the package's performance in their own IT infrastructure.

7 Call for Contributors

The task at hand, although interesting, is complex and does require the input and expertise of a team. At present, that team is a team of one. So, this article is not only to introduce this package and gain traction but also a call for potential contributors. You need not have Python expertise to be part of the team. Please reach out to the author if you want to be part of this team.