# aws for you

## Instance in an Instant

# The Problem

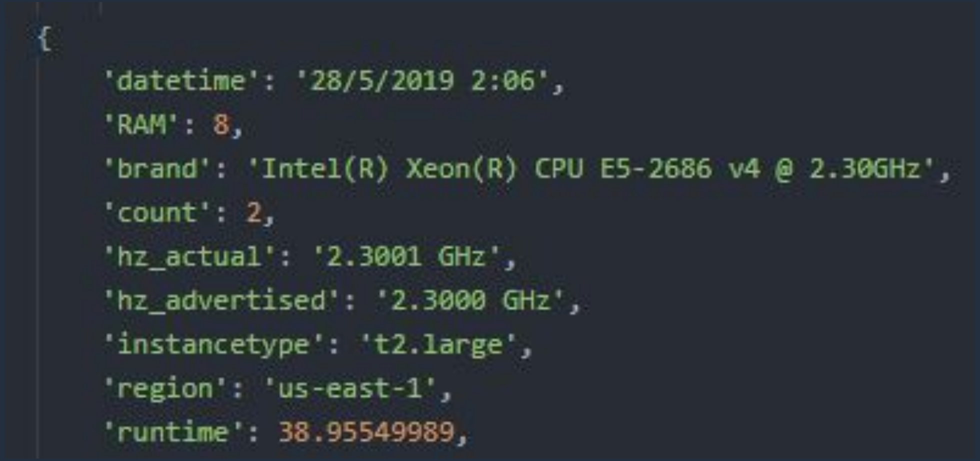
- Students/researchers in data science have no concrete way to choose AWS instance type.
  - A1, T3, T3a, T2, M5, M5a, M4, C5, C5n, C4
- There is computation time vs cost tradeoff.

# The Solution

- Estimate how long the user’s algorithm will take to run on various AWS instances.
- Suggest the best instance choice for runtime or cost.

# The Data

- AWS performance data
    - Generated by running benchmark on various types of AWS instances



```
{
        'datetime': '28/5/2019 2:06',
        'RAM': 8,
        'brand': 'Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz',
        'count': 2,
        'hz_actual': '2.3001 GHz',
        'hz_advertised': '2.3000 GHz',
        'instancetype': 't2.large',
        'region': 'us-east-1',
        'runtime': 38.95549989,
```

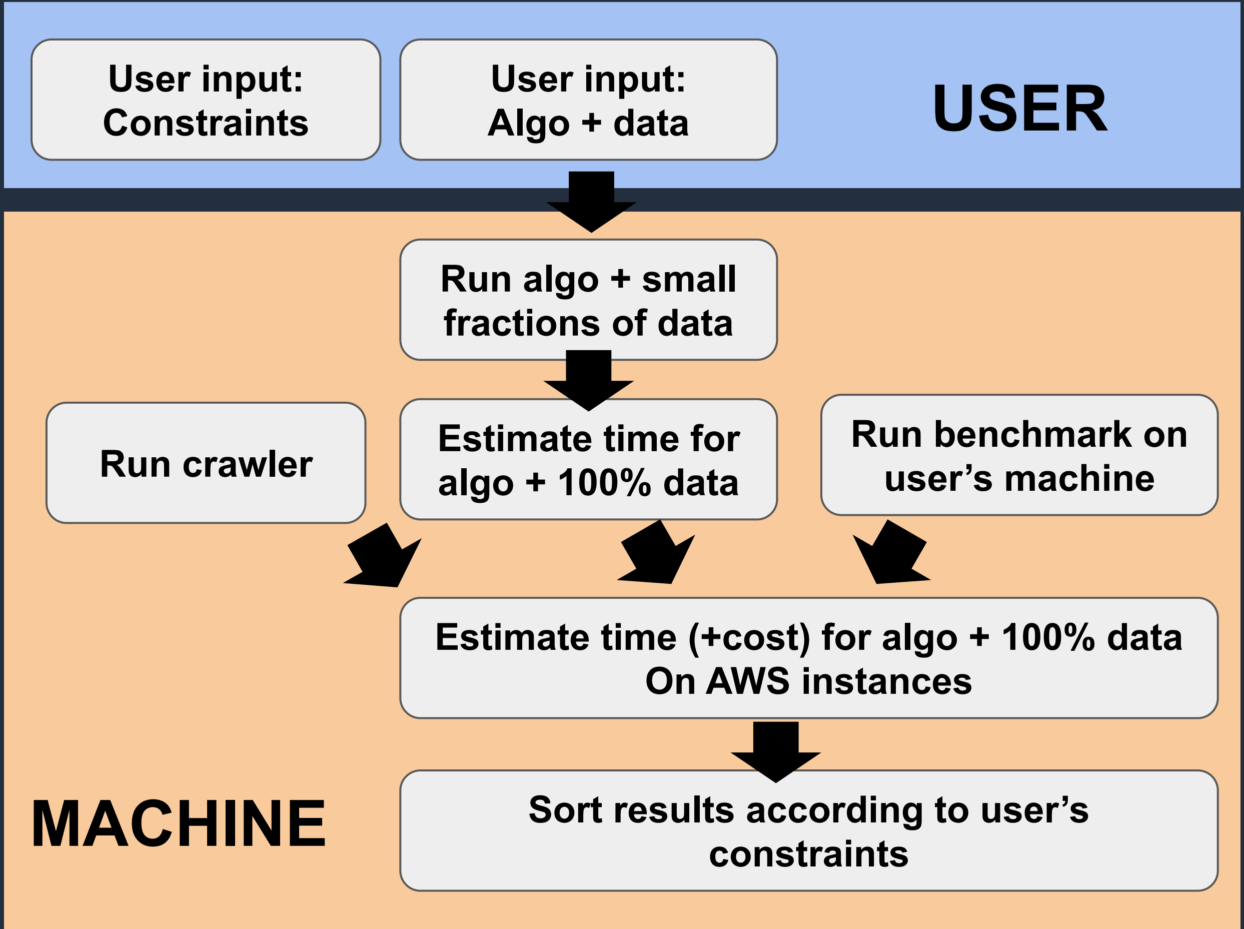
    - Limitation: New type of AWS instance has to be benchmarked

- On-demand and spot price data
    - Generated by a crawler

# Use cases

- Choosing the fastest instance given budget.

- Choosing the cheapest instance given time.



USER

User input: Constraints

User input: Algo + data

MACHINE

Run algo + small fractions of data

Run crawler

Estimate time for algo + 100% data

Run benchmark on user’s machine

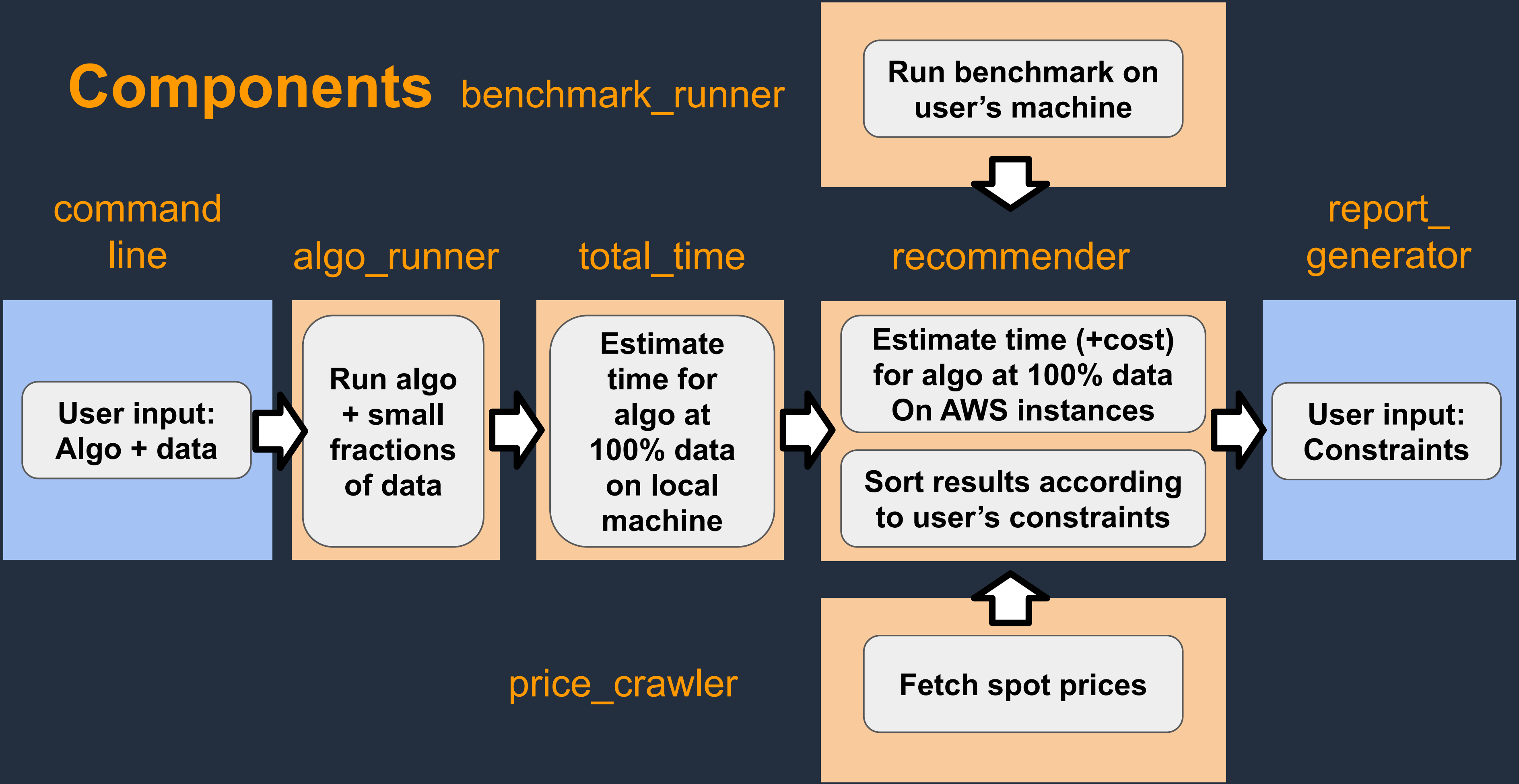
Estimate time (+cost) for algo + 100% data On AWS instances

Sort results according to user’s constraints

# demo

Instance in an Instant

# Components



benchmark_runner

Run benchmark on user's machine

command line

User input: Algo + data

algo_runner

Run algo + small fractions of data

total_time

Estimate time for algo at 100% data on local machine

recommender

Estimate time (+cost) for algo at 100% data On AWS instances

Sort results according to user's constraints

report_ generator

User input: Constraints

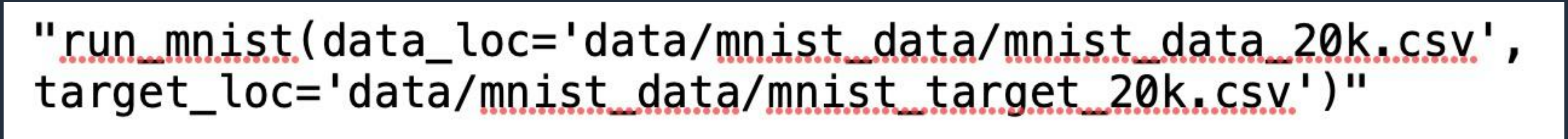
price_crawler

Fetch spot prices

# algo_runner

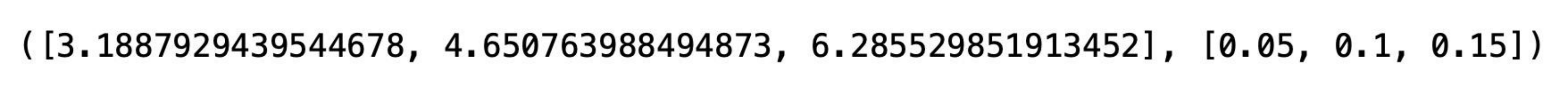
Run algo + small fractions of data

- Takes as inputs the python used to call the code in question:



```
"run_mnist(data_loc='data/mnist_data/mnist_data_20k.csv',
target_loc='data/mnist_data/mnist_target_20k.csv')"
```

- Outputs two lists, one containing the percentage of samples run per iteration, the other the amount of time (in seconds) it took to run each of the three iterations:



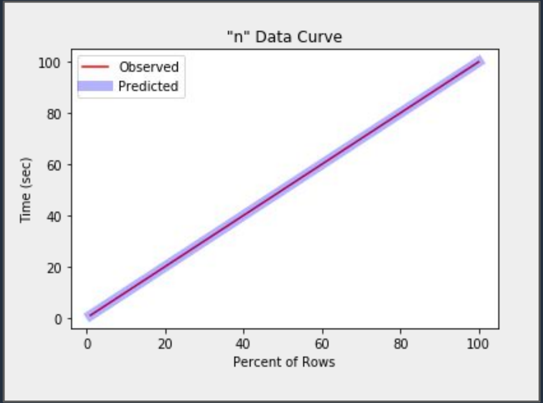
```
([3.1887929439544678, 4.650763988494873, 6.285529851913452], [0.05, 0.1, 0.15])
```

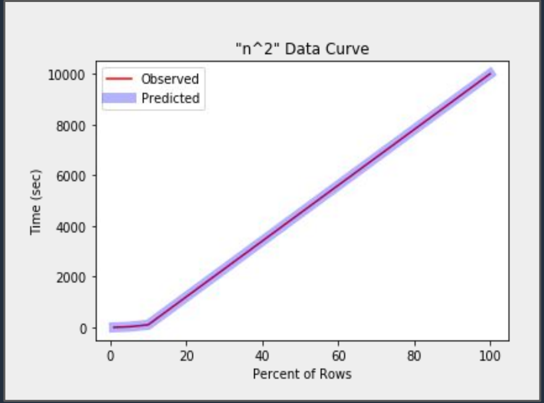
# total_time

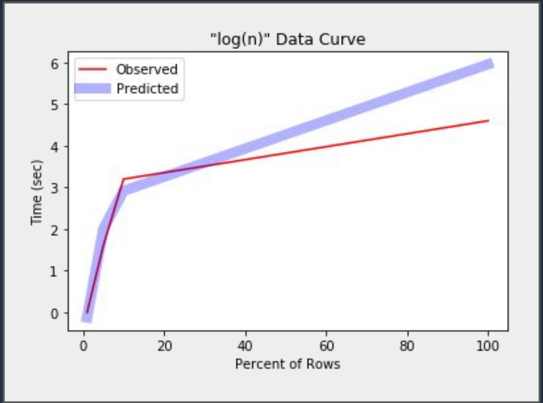
**Estimate time for algo at 100% data on local machine**

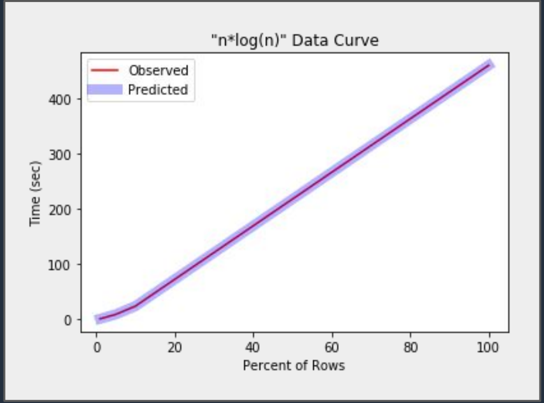
We need to extrapolate the algo runtime at 100% data i.e. estimating complexity.

O(n, $n^2$, 1/n, nlog(n)).


"n" Data Curve


"n^2" Data Curve

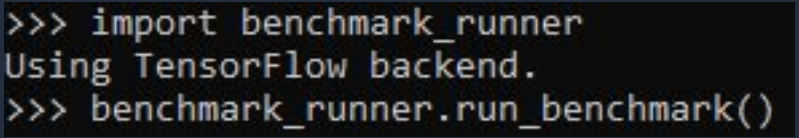

"log(n)" Data Curve


"n*log(n)" Data Curve

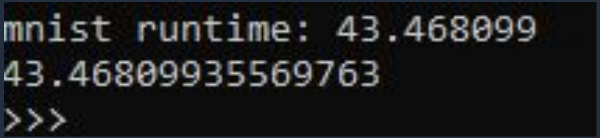
# benchmark_runner

**Run benchmark on user's machine**

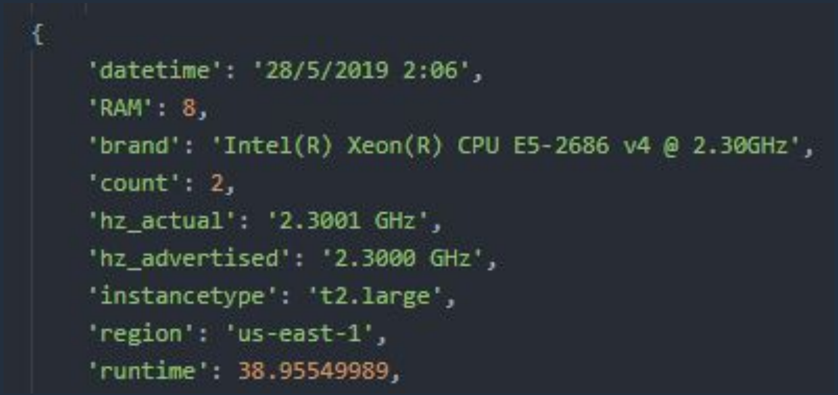
- Component that runs benchmark on user's machine.
  - Trains a classifier on mnist data (image recognition of digits from 0-9)
  - 60,000 rows for training set, 10,000 rows for test set.
- Also used to run and record the runtime on various AWS instances.



```
>>> import benchmark_runner
Using TensorFlow backend.
>>> benchmark_runner.run_benchmark()
```



```
mnist runtime: 43.468099
43.46809935569763
>>>
```

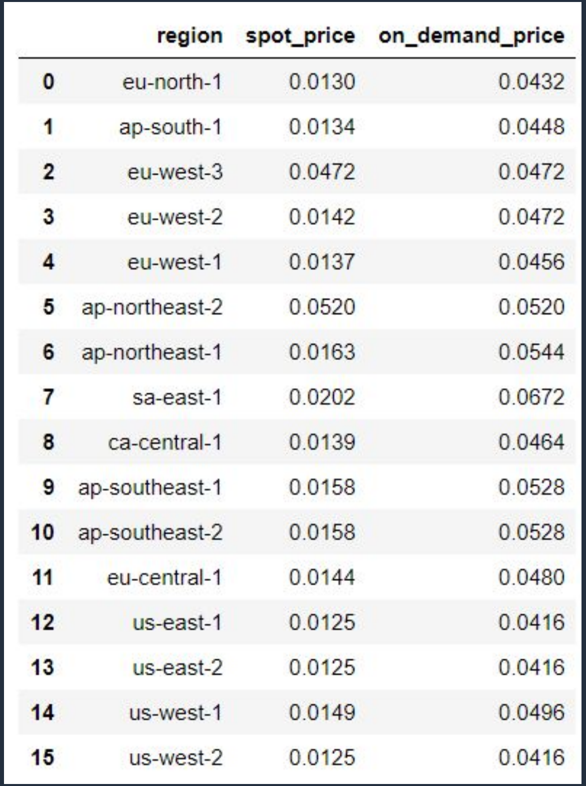


```
{
    'datetime': '28/5/2019 2:06',
    'RAM': 8,
    'brand': 'Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz',
    'count': 2,
    'hz_actual': '2.3001 GHz',
    'hz_advertised': '2.3000 GHz',
    'instancetype': 't2.large',
    'region': 'us-east-1',
    'runtime': 38.95549989,
```

# price_crawler

Fetch spot prices

- Component that fetches the spot price and on-demand price of a given instance at the current instant as a pandas dataframe
  - Get spot price using boto3
  - Get on-demand price using pre aggregated database



| | region | spot_price | on_demand_price |
|---|---|---|---|
| **0** | eu-north-1 | 0.0130 | 0.0432 |
| **1** | ap-south-1 | 0.0134 | 0.0448 |
| **2** | eu-west-3 | 0.0472 | 0.0472 |
| **3** | eu-west-2 | 0.0142 | 0.0472 |
| **4** | eu-west-1 | 0.0137 | 0.0456 |
| **5** | ap-northeast-2 | 0.0520 | 0.0520 |
| **6** | ap-northeast-1 | 0.0163 | 0.0544 |
| **7** | sa-east-1 | 0.0202 | 0.0672 |
| **8** | ca-central-1 | 0.0139 | 0.0464 |
| **9** | ap-southeast-1 | 0.0158 | 0.0528 |
| **10** | ap-southeast-2 | 0.0158 | 0.0528 |
| **11** | eu-central-1 | 0.0144 | 0.0480 |
| **12** | us-east-1 | 0.0125 | 0.0416 |
| **13** | us-east-2 | 0.0125 | 0.0416 |
| **14** | us-west-1 | 0.0149 | 0.0496 |
| **15** | us-west-2 | 0.0125 | 0.0416 |

# recommender

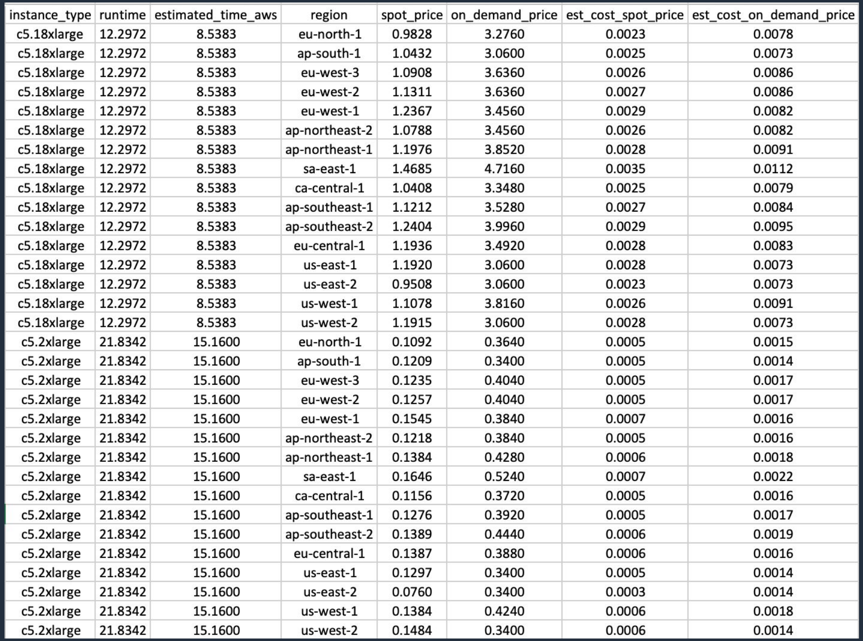
Estimate time (+cost) for algo at 100% data On AWS instances

Sort results according to user's constraints

Calls:

- algo_runner
- algo_analyzer
- benchmark_runner
- price_crawler

Outputs a dataframe to the "report_generator" component, that has total estimated times, total estimated costs for each instance type.



| instance_type | runtime | estimated_time_aws | region | spot_price | on_demand_price | est_cost_spot_price | est_cost_on_demand_price |
|---|---|---|---|---|---|---|---|
| c5.18xlarge | 12.2972 | 8.5383 | eu-north-1 | 0.9828 | 3.2760 | 0.0023 | 0.0078 |
| c5.18xlarge | 12.2972 | 8.5383 | ap-south-1 | 1.0432 | 3.0600 | 0.0025 | 0.0073 |
| c5.18xlarge | 12.2972 | 8.5383 | eu-west-3 | 1.0908 | 3.6360 | 0.0026 | 0.0086 |
| c5.18xlarge | 12.2972 | 8.5383 | eu-west-2 | 1.1311 | 3.6360 | 0.0027 | 0.0086 |
| c5.18xlarge | 12.2972 | 8.5383 | eu-west-1 | 1.2367 | 3.4560 | 0.0029 | 0.0082 |
| c5.18xlarge | 12.2972 | 8.5383 | ap-northeast-2 | 1.0788 | 3.4560 | 0.0026 | 0.0082 |
| c5.18xlarge | 12.2972 | 8.5383 | ap-northeast-1 | 1.1976 | 3.8520 | 0.0028 | 0.0091 |
| c5.18xlarge | 12.2972 | 8.5383 | sa-east-1 | 1.4685 | 4.7160 | 0.0035 | 0.0112 |
| c5.18xlarge | 12.2972 | 8.5383 | ca-central-1 | 1.0408 | 3.3480 | 0.0025 | 0.0079 |
| c5.18xlarge | 12.2972 | 8.5383 | ap-southeast-1 | 1.1212 | 3.5280 | 0.0027 | 0.0084 |
| c5.18xlarge | 12.2972 | 8.5383 | ap-southeast-2 | 1.2404 | 3.9960 | 0.0029 | 0.0095 |
| c5.18xlarge | 12.2972 | 8.5383 | eu-central-1 | 1.1936 | 3.4920 | 0.0028 | 0.0083 |
| c5.18xlarge | 12.2972 | 8.5383 | us-east-1 | 1.1920 | 3.0600 | 0.0028 | 0.0073 |
| c5.18xlarge | 12.2972 | 8.5383 | us-east-2 | 0.9508 | 3.0600 | 0.0023 | 0.0073 |
| c5.18xlarge | 12.2972 | 8.5383 | us-west-1 | 1.1078 | 3.8160 | 0.0026 | 0.0091 |
| c5.18xlarge | 12.2972 | 8.5383 | us-west-2 | 1.1915 | 3.0600 | 0.0028 | 0.0073 |
| c5.2xlarge | 21.8342 | 15.1600 | eu-north-1 | 0.1092 | 0.3640 | 0.0005 | 0.0015 |
| c5.2xlarge | 21.8342 | 15.1600 | ap-south-1 | 0.1209 | 0.3400 | 0.0005 | 0.0014 |
| c5.2xlarge | 21.8342 | 15.1600 | eu-west-3 | 0.1235 | 0.4040 | 0.0005 | 0.0017 |
| c5.2xlarge | 21.8342 | 15.1600 | eu-west-2 | 0.1257 | 0.4040 | 0.0005 | 0.0017 |
| c5.2xlarge | 21.8342 | 15.1600 | eu-west-1 | 0.1545 | 0.3840 | 0.0007 | 0.0016 |
| c5.2xlarge | 21.8342 | 15.1600 | ap-northeast-2 | 0.1218 | 0.3840 | 0.0005 | 0.0016 |
| c5.2xlarge | 21.8342 | 15.1600 | ap-northeast-1 | 0.1384 | 0.4280 | 0.0006 | 0.0018 |
| c5.2xlarge | 21.8342 | 15.1600 | sa-east-1 | 0.1646 | 0.5240 | 0.0007 | 0.0022 |
| c5.2xlarge | 21.8342 | 15.1600 | ca-central-1 | 0.1156 | 0.3720 | 0.0005 | 0.0016 |
| c5.2xlarge | 21.8342 | 15.1600 | ap-southeast-1 | 0.1276 | 0.3920 | 0.0005 | 0.0017 |
| c5.2xlarge | 21.8342 | 15.1600 | ap-southeast-2 | 0.1389 | 0.4440 | 0.0006 | 0.0019 |
| c5.2xlarge | 21.8342 | 15.1600 | eu-central-1 | 0.1387 | 0.3880 | 0.0006 | 0.0016 |
| c5.2xlarge | 21.8342 | 15.1600 | us-east-1 | 0.1297 | 0.3400 | 0.0005 | 0.0014 |
| c5.2xlarge | 21.8342 | 15.1600 | us-east-2 | 0.0760 | 0.3400 | 0.0003 | 0.0014 |
| c5.2xlarge | 21.8342 | 15.1600 | us-west-1 | 0.1384 | 0.4240 | 0.0006 | 0.0018 |
| c5.2xlarge | 21.8342 | 15.1600 | us-west-2 | 0.1484 | 0.3400 | 0.0006 | 0.0014 |

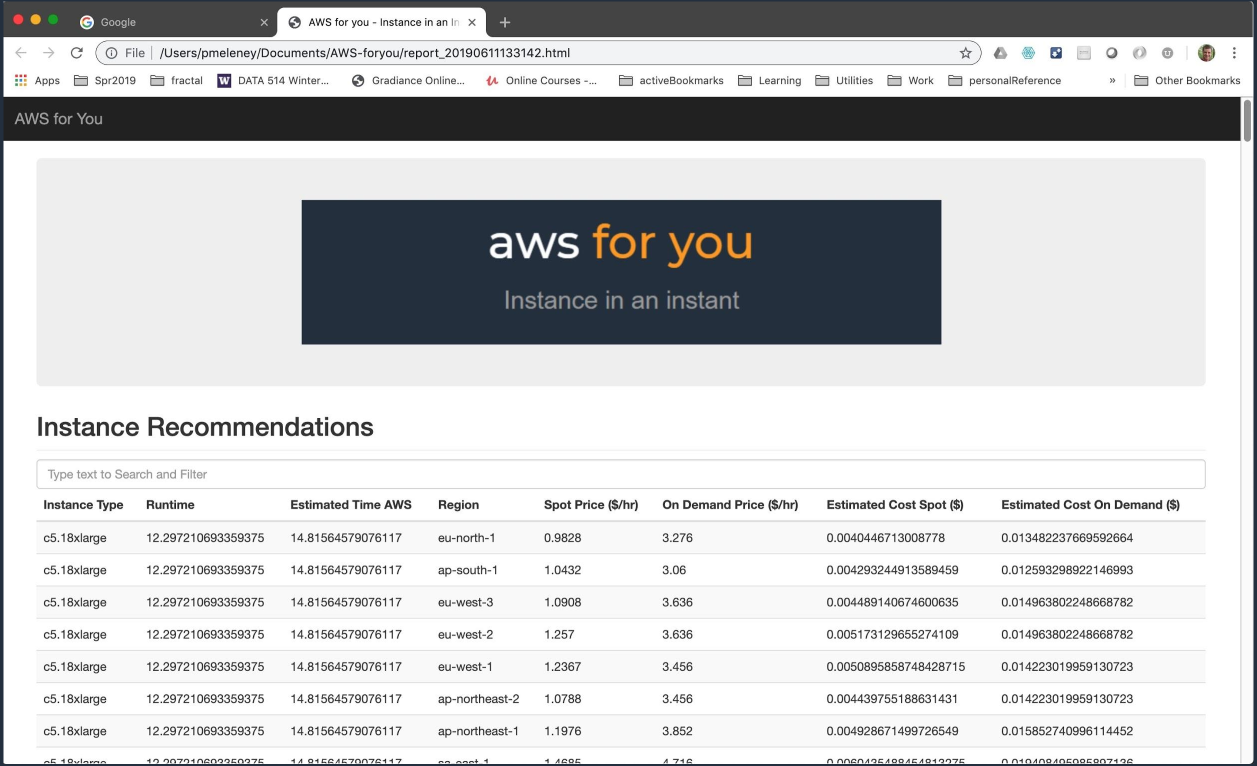
# report_generator

User input: Constraints

User input: Algo + data

Takes the dataframe output from the recommender and generates a sortable html table

User finds best instance for price or time by sorting



AWS for You

aws for you

Instance in an instant

## Instance Recommendations

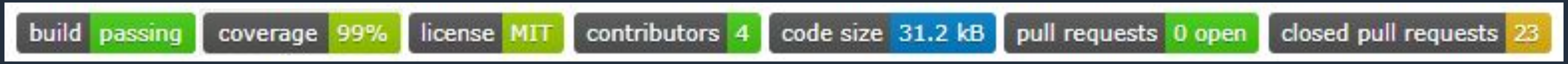
Type text to Search and Filter

| Instance Type | Runtime | Estimated Time AWS | Region | Spot Price ($/hr) | On Demand Price ($/hr) | Estimated Cost Spot ($) | Estimated Cost On Demand ($) |
|---|---|---|---|---|---|---|---|
| c5.18xlarge | 12.297210693359375 | 14.81564579076117 | eu-north-1 | 0.9828 | 3.276 | 0.0040446713008778 | 0.013482237669592664 |
| c5.18xlarge | 12.297210693359375 | 14.81564579076117 | ap-south-1 | 1.0432 | 3.06 | 0.004293244913589459 | 0.012593298922146993 |
| c5.18xlarge | 12.297210693359375 | 14.81564579076117 | eu-west-3 | 1.0908 | 3.636 | 0.004489140674600635 | 0.014963802248668782 |
| c5.18xlarge | 12.297210693359375 | 14.81564579076117 | eu-west-2 | 1.257 | 3.636 | 0.005173129655274109 | 0.014963802248668782 |
| c5.18xlarge | 12.297210693359375 | 14.81564579076117 | eu-west-1 | 1.2367 | 3.456 | 0.0050895858748428715 | 0.014223019959130723 |
| c5.18xlarge | 12.297210693359375 | 14.81564579076117 | ap-northeast-2 | 1.0788 | 3.456 | 0.004439755188631431 | 0.014223019959130723 |
| c5.18xlarge | 12.297210693359375 | 14.81564579076117 | ap-northeast-1 | 1.1976 | 3.852 | 0.004928671499726549 | 0.015852740996114452 |
| c5.18xlarge | 12.297210693359375 | 14.81564579076117 | sa-east-1 | 1.4685 | 4.716 | 0.0060435488454813275 | 0.019408495985897136 |

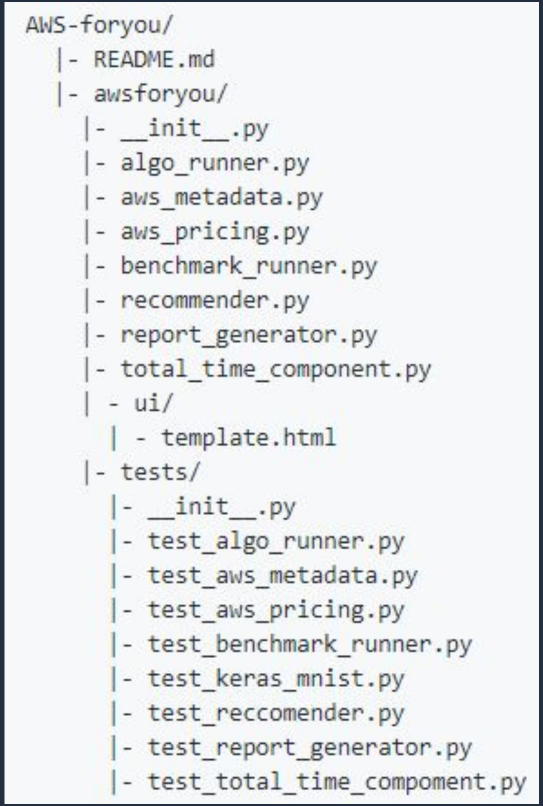
# demo

Instance in an Instant

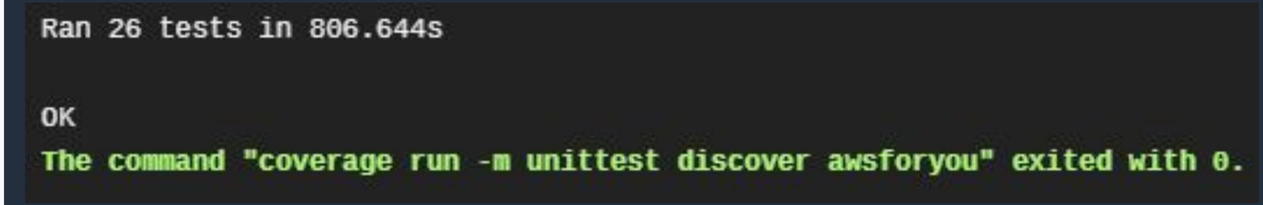
# Project Structure and Continuous Integration



build passing | coverage 99% | license MIT | contributors 4 | code size 31.2 kB | pull requests 0 open | closed pull requests 23

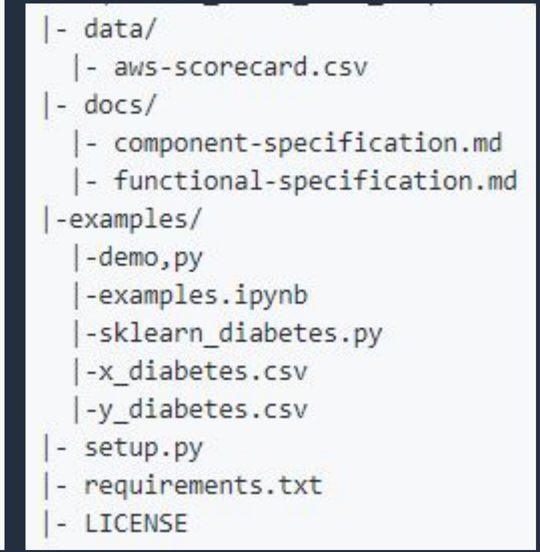


```
AWS-foryou/
  |- README.md
  |- awsforyou/
    |- __init__.py
    |- algo_runner.py
    |- aws_metadata.py
    |- aws_pricing.py
    |- benchmark_runner.py
    |- recommender.py
    |- report_generator.py
    |- total_time_component.py
    |  - ui/
      | - template.html
    |- tests/
      |- __init__.py
      |- test_algo_runner.py
      |- test_aws_metadata.py
      |- test_aws_pricing.py
      |- test_benchmark_runner.py
      |- test_keras_mnist.py
      |- test_reccomender.py
      |- test_report_generator.py
      |- test_total_time_compoment.py
```



```
Ran 26 tests in 806.644s

OK
The command "coverage run -m unittest discover awsforyou" exited with 0.
```



```
|- data/
  |- aws-scorecard.csv
|- docs/
  |- component-specification.md
  |- functional-specification.md
|-examples/
  |-demo,py
  |-examples.ipynb
  |-sklearn_diabetes.py
  |-x_diabetes.csv
  |-y_diabetes.csv
|- setup.py
|- requirements.txt
|- LICENSE
```

# Challenges

- Configuring security credentials in CI
- Correct model selection for time estimate
- AWS python SDK (boto3) API
- Writing tests for code meant to be run on AWS to fetch instance-related metadata (mocking).
- pip installable package

# Future Work

- Ability to recommend GPU instances
- Extend coverage to Azure/Google Cloud

# aws for you

Instance in an instant