Warehouse Storage Optimization - Report 4

Dhruvil Dave, Dhatri Kapuriya, Harvish Jariwala, Nisarg Thoriya
School of Engineering and Applied Sciences
Ahmedabad University
Ahmedabad, Gujarat, India - 380009
dhruvil.d@ahduni.edu.in (AU1841003)
dhatri.k@ahduni.edu.in (AU1841129)
harvish.j@ahduni.edu.in (AU1841050)
nisarg.t@ahduni.edu.in (AU1841142)

***Abstract*—This is the fourth progress report of our group *Gopher - Group 4* for Machine Learning (CSE523) course project.**

***Index Terms*—Time series forecasting, classification, data pre-processing, clustering, graphical models**

I. Introduction

For our project, we decided to use the Amazon Bin Images Dataset.

This is originally a Computer Vision dataset. The Amazon Bin Image Dataset contains over 530,000 images and metadata from bins of a pod in an operating Amazon Fulfillment Center.

II. Brief insight of previous work

As we decided to use Time series forecasting in the previous week, we came up with the modelling of the situation and basic working flow of algorithm.

III. Task Performed And Outcomes

In this week we created a synthetic data and tried to visualise the scenario where products arrive at any point of time further predicting the possible way in which they will be placed in the bins. Initially, assuming that there are 300 bins present in the warehouse and all of them are empty. Then we create a random data of volumes for 15000 products. The product is placed in the bin which has capacity to accommodate the product.At every iteration bins are sorted on the basis of empty space available. This optimises the problem and tries to accommodate the products in least remaining space. Later we created a dataframe of the the output obtained at every iteration of the data.

```
import json
import numpy as np
from tqdm import tqdm

def main() -> None:
    arr = []
    bins = dict.fromkeys(range(200), 1)
    vol = np.random.rand(10000)
    for i in tqdm(vol):
        placed = True
        for j in bins:
            if i < bins[j]:
                bins[j] -= i
                break
        else:
            placed = False

        bins = dict(sorted(bins.items(), key=lambda item: item[1]))

        arr.append((i, bins, placed))
    json.dump(arr, open("ds.json", "w"))

if __name__ == "__main__":
    main()
```

In the following screenshot, we have displayed our synthetic data set so far. We have scaled the previous algorithm to 15000 items and 300 bins.

- In it, the first column indicates the volume of the product that will be placed into the bins in this iteration.
- The second column portrays the change in state every time the product is placed into the bin. We can see that when the 3rd object is placed into the bin, the volume of 0th bin changes from 0.44219 to 0.17206 (Note: The number indicates the fraction of bin that is empty as discussed in the previous report).
- Finally, the last column is used to depict whether the product is places in the bin at all or not (True value indicates the product is placed, false value indicates there was no empty space in the bins at all to place the new product).

	0	1	2
0	0.557809	{'0': 0.442190832743638, '1': 0.22276782926915...	True
1	0.777232	{'1': 0.22276782926915803, '0': 0.442190832743...	True
2	0.531790	{'1': 0.22276782926915803, '0': 0.172067432700...	True
3	0.270123	{'0': 0.172067432700429, '1': 0.22276782926915...	True
4	0.897342	{'3': 0.102657903071192, '0': 0.172067432700042...	True
...	...	...	...
14995	0.573324	{'268': 3.0945429152495757e-07, '285': 3.33242...	False
14996	0.693822	{'268': 3.0945429152495757e-07, '285': 3.33242...	False
14997	0.577438	{'268': 3.0945429152495757e-07, '285': 3.33242...	False
14998	0.723491	{'268': 3.0945429152495757e-07, '285': 3.33242...	False
14999	0.218421	{'268': 3.0945429152495757e-07, '285': 3.33242...	False

[15000 rows x 3 columns]

IV. Tasks for Upcoming Week

The main tasks to be performed in the upcoming week are:

1) Training models
2) Tuning models
3) Adapting the model to dataset