Warehouse Storage Optimization - Report 5

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 5* 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

After coming up with the modelling of the problem, we generated synthetic data last week. We generated the data similar to the Amazon bin images dataset. On every iteration i.e. on every new row, a product was being added to a list of 10 bins and then the bins were sorted everytime to maintain the bin order. Using the python code, we generated a dataset of 300 bins and 15,000 products

III. Task Performed And Outcomes

In this week, we expanded our dataset in tabular form i.e till last week we had data column in the form of dictionary which was spreaded out into columns of separate bins.

While expanding the data we observed a bug in the dataset we create previous week. In it, our requirement was that each row must represent a volume of input product and the bin state after the product was placed. But, instead our dataset produced results such as at every row the sate of bins gets updated. Which implies that if any product is placed in the bin, then the change will be reflected in the bin state of previous rows too, which was not the likely requirement. Later by making some changes in the script we achieved desired results.

```python
#!/usr/bin/env python3

import asyncio
```

```python
import json

import numpy as np
from tqdm import tqdm


def main() -> None:
    for k in tqdm(range(5000)):
        np.random.seed(k)
        arr = []
        bins = dict.fromkeys(range(20), 1)
        vol = np.random.rand(200)
        for i in vol:
            placed = True
            for j in bins:
                if i < bins[j]:
                    bins[j] -= i
                    bin_no = j
                    break
            else:
                placed = False

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

            arr.append((i, bins.copy(), placed,
    bin_no))
        json.dump(arr, open(f"data/{k}.json", "w"))


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

The bins in the data were present in a dictionary form and we tabularized to normal form as seen in the image:

	0	2	3	100	101	102	103	104	105	106	...	110	111	112	113	114	115
0	0.208461	True	0	0.791539	1.000000	1.000000	1.00000	1.00000	1.000000	1.0000	...	1.000000	1.000000	1.00000	1.000000	1.000000	1.000000
1	0.481681	True	0	0.309858	1.000000	1.000000	1.00000	1.00000	1.000000	1.0000	...	1.000000	1.000000	1.00000	1.000000	1.000000	1.000000
2	0.420538	True	1	0.309858	0.579462	1.000000	1.00000	1.00000	1.000000	1.0000	...	1.000000	1.000000	1.00000	1.000000	1.000000	1.000000
3	0.859182	True	2	0.309858	0.579462	0.140818	1.00000	1.00000	1.000000	1.0000	...	1.000000	1.000000	1.00000	1.000000	1.000000	1.000000
4	0.171162	True	0	0.138697	0.579462	0.140818	1.00000	1.00000	1.000000	1.0000	...	1.000000	1.000000	1.00000	1.000000	1.000000	1.000000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
999995	0.315423	False	13	0.003609	0.034373	0.002928	0.00008	0.00027	0.013412	0.0038	...	0.023717	0.016353	0.05163	0.003678	0.002267	0.001295
999996	0.230940	False	13	0.003609	0.034373	0.002928	0.00008	0.00027	0.013412	0.0038	...	0.023717	0.016353	0.05163	0.003678	0.002267	0.001295
999997	0.933347	False	13	0.003609	0.034373	0.002928	0.00008	0.00027	0.013412	0.0038	...	0.023717	0.016353	0.05163	0.003678	0.002267	0.001295
999998	0.971583	False	13	0.003609	0.034373	0.002928	0.00008	0.00027	0.013412	0.0038	...	0.023717	0.016353	0.05163	0.003678	0.002267	0.001295
999999	0.988404	False	13	0.003609	0.034373	0.002928	0.00008	0.00027	0.013412	0.0038	...	0.023717	0.016353	0.05163	0.003678	0.002267	0.001295

1000000 rows × 23 columns

Last week we had 15000 rows and we created more data using sythetic data generatioin libraries like CTGAN. This gave us 1 million rows for 20 bins.

This week, we also trained a basic Decision tree classifier on the generated data. We split the data into a 95% train and 5% test set. Which gives us around 50,000 rows for the testing dataset. Our output column is '3'. Score of 89.43%

```
In [24]: model.score(test_x, test_y)
Out[24]: 0.89432
```

IV. Tasks for Upcoming Week

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

1) Modelling
2) Minor tweaking for modelling