FRANCIUM TECH

Customer Segmentation With Google Dataflow and TensorFlow

Premise

Identifying customer's buying habit and categorizing them based on their shopping history is a key in retail domain today. It allows the companies to do better promotion of products and improves sales. One of our clients wanted to do user-segmentation based on their purchase history over the past one year and based on the outcome they wanted to release promotional offers suiting each segment.

The main challenge we faced here was the volume of data. The system we design should handle the huge amount of data which had already been accumulated so far. Another important thing is that the system should be able to scale for a stream of real time sales data. Keeping these two in mind, we chose Google Cloud Platform(GCP) to do this.

Google Cloud Platform

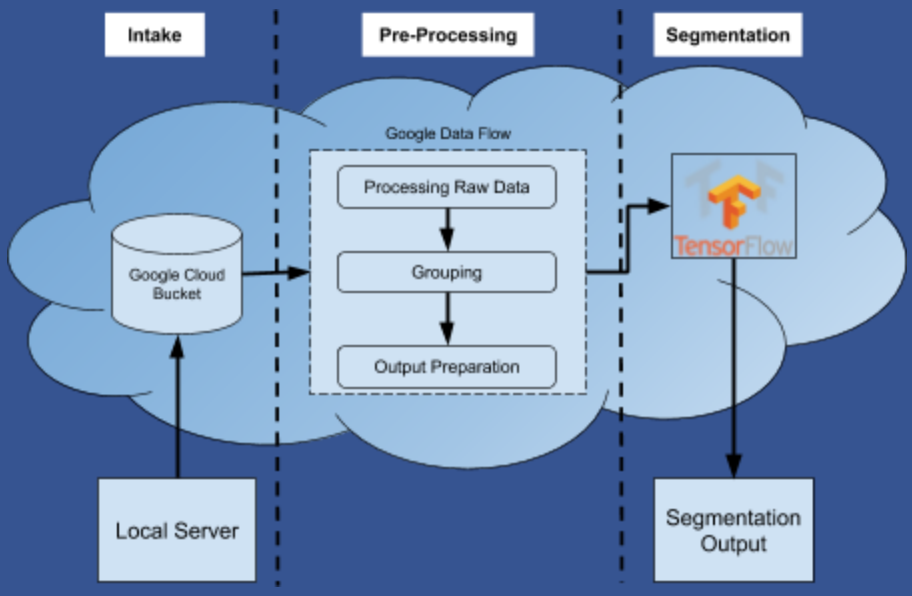
Google cloud platform offers all the essential pieces for our task. The advantages of going with GCP are as follows

- **Serverless Model** - Google DataFlow is a serverless data processing pipeline, which means we only have to concentrate on the data processing logic and the operations part will be taken care automatically by the GCP.
- **Auto Scaling** - Based on the data load it can scale up/down the servers needed.
- **Google Ecosystem** - Google provides all the necessary tools and APIs for all the relevant data processings.

The Process

The system we designed was a three step process based.

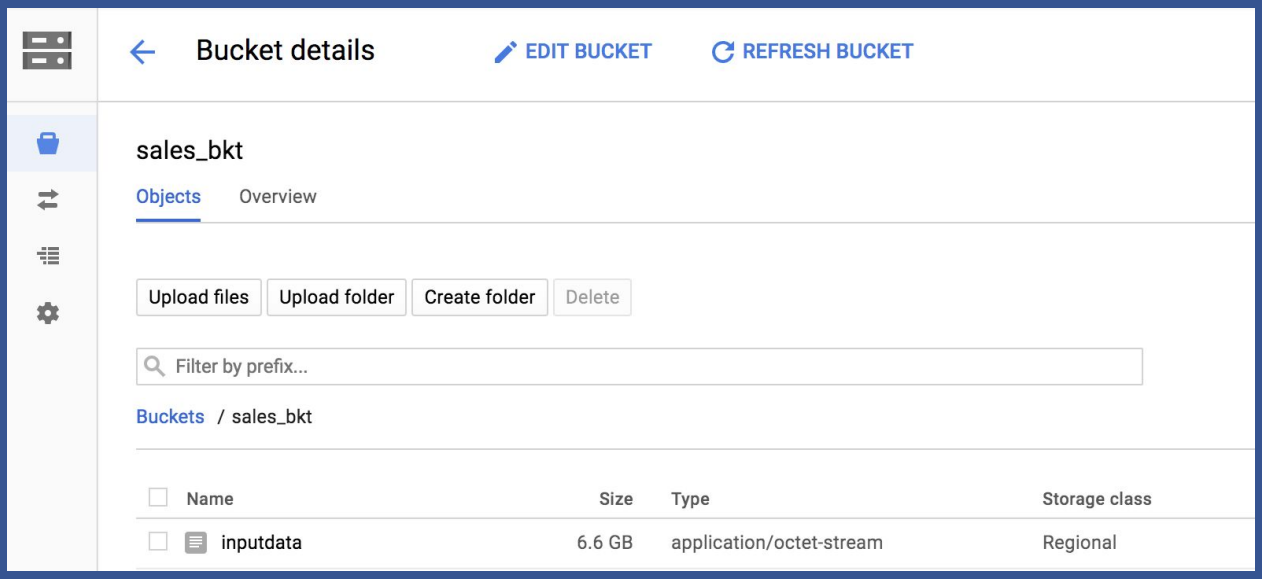
1. Intake
2. Pre-Processing
3. Segmentation



Intake

As we already had mentioned, the data we were dealing with was huge, and we decided to go with cloud-processing. Practically holding the data in local or any server, and inputting them directly onto a cloud-processing system is not appropriate. To solve this, we decided to upload the historical sales data into our cloud-storage- bucket first.

Once the uploading was over, we were able to check our data in the cloud bucket. In our case, the file(inputdata) we uploaded was 6.6 GB in size.



Pre-Processing

The data we had uploaded to google-cloud was raw. We had to make it consumable for our clustering, and that was exactly where the DataFlow came in. In this implementation we used python SDK to work with the DataFlow.

We needed the following libraries to get started

- argparsing
- logging
- re
- six
- Apache Beam
- datetime
- json

Pipeline Code Snippet

```
with beam.Pipeline(options=pipeline_options) as p:
    output = (p | ReadFromText(known_args.input)
    | beam.ParDo(ParseRecordDoFnUsrClust())
    | beam.GroupByKey()
    | beam.ParDo(ParseRecordDoFnmap()))
    output | WriteToText(known_args.output)
```

Extraction:

The raw sales records we had was at product and user level, we had records for each product and user, so our first ParDo function should extract userid and the price of the product purchased.

Grouping

The next ParDo function should get the grouped data as input and compute the number of products purchased and the price paid in total at user level and finally wrote the result into output files.

Once we ran this, we were able to see our job running on cloud. Here is how our pipeline looked like in the console.

If you notice the Elapsed Time, it took was just 18 minutes to process 6.6GB data with just 8 workers. The output of our pipeline was stored in cloud storage itself, we then downloaded it again to use as a input to our TensorFlow model. The output was written into multiple files(as default) as the pipeline was processing the data over multiple servers in parallel. We wrote a simple code to download these files and concatenate them into a single file for our TensorFlow model.

Segmentation

Next step was to start building the clustering model in TensorFlow to segment the customers.

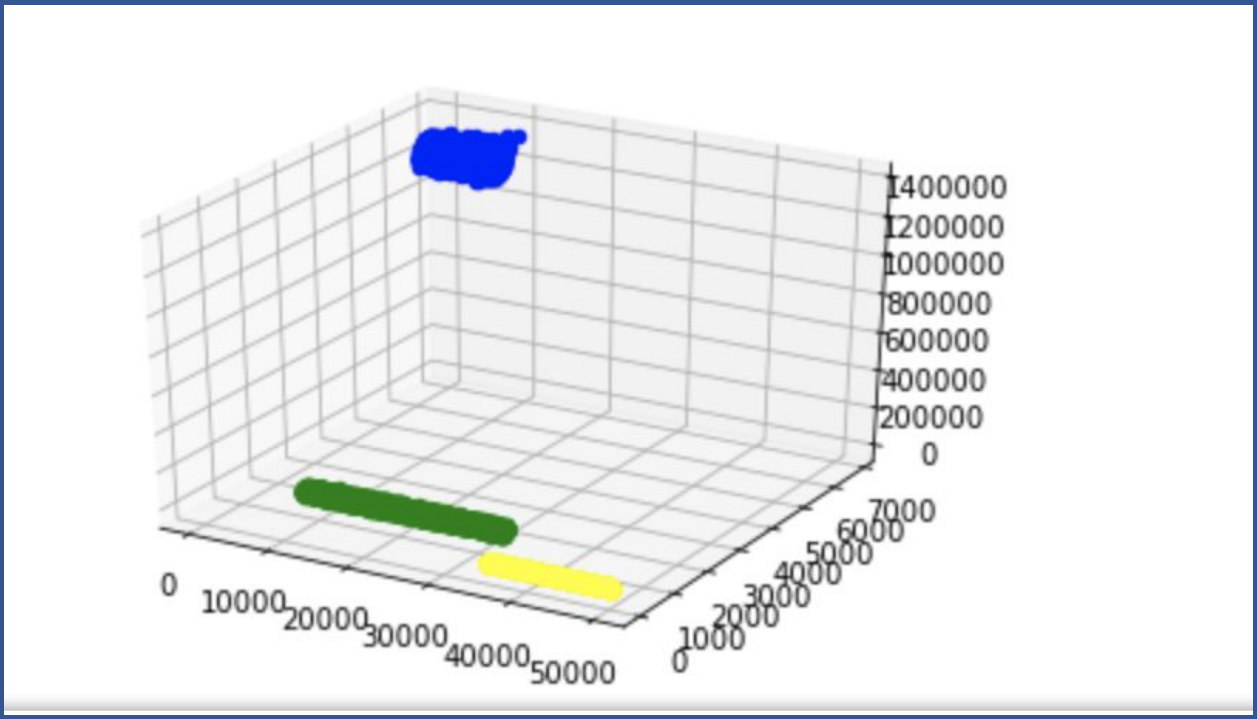
The model is supposed to segment the users into three different classes High, Medium, Low. We used fairly simple **K-means clustering** model here.

```
def train_data():
  ip_dt = tf.constant(ip_data, tf.float32)
  return (ip_dt, None)

def pred_fn():
  return np.array(ip_data, np.float32)
  model = tf.contrib.learn.KMeansClustering(classes,
        distance_metric = clustering_ops.SQUARED_EUCLIDEAN_DISTANCE,
        initial_clusters=tf.contrib.learn.KMeansClustering.RANDOM_INIT )

model.fit(input_fn=train_data, steps=5000)
preds = model.predict(input_fn=pred_fn, as_iterable=True)
```

At the end of the execution we were able to visualize the user segmentation done based on the data we processed in DataFlow and it looked like below.



Conclusion

Three classes of customer base is very obvious in our dataset based on the customer contribution towards sales. Now the company can release the promotional offers personalised to each customer group. We can use the same technique to segment the customer based on various other attributes like the product group they are buying mostly and customer behaviour during the promotional sales and ordinary days.