# Title: Deal insights - March 2019

## Team:

**ML Engineering**: Sarthak Dev

**FS Engineering**: Sivalingam, Mudhabir

**Data Science**: Sai Charan, Jagadeesh

**Leads**: Swami, Sudharshan, Srivatsan

**PM**: Aditi Balaji

**UX**: Arun, Pawan

## Problem Statement:

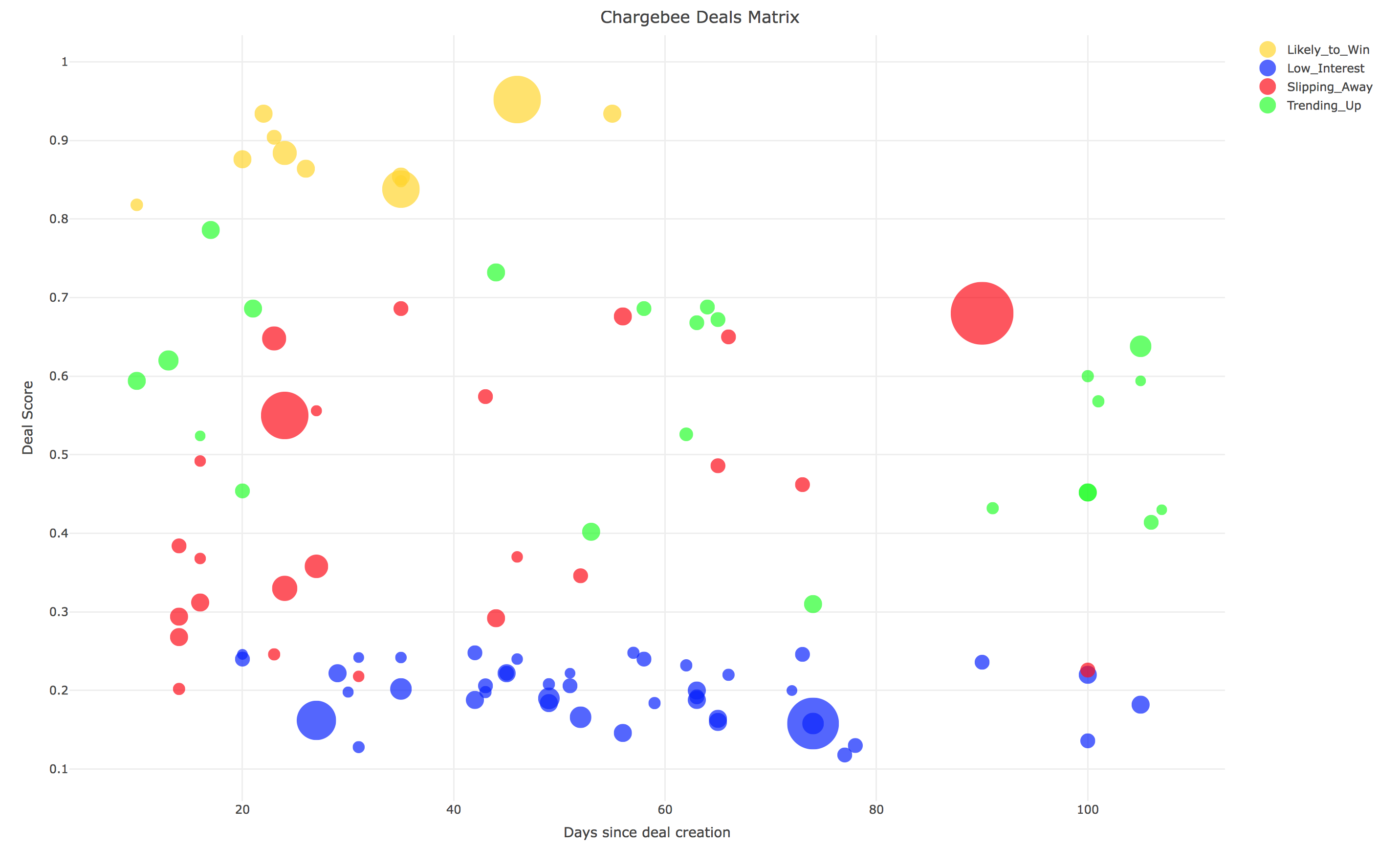
To TAG deals as "Trending UP" or "Slipping Down", "Likely to WIN" etc.. and also give [Causal](https://en.wikipedia.org/wiki/Causal_inference) reasoning for why the deal is "Trending UP" or "Slipping Down". Additionally, we must also identify the context in which we should show/not show the TAG and Signals. Thus deal insights is a layer on TOP of the predictive deal scoring feature (the earlier scope).

## Learnings from chargebee account:

* We are live for chargebee (currently we push the deal insights as TAGs using their API after giving a demo to their sales team)
* Customers seems to need more detailed communication related to how we are creating different TAGs and what are the key insights for their account   
  → Educating the customers is going to be an important factor for the success of this feature

## Deals Matrix (POC for chargebee):

This shall help the sales teams in their Sales Pipeline reviews and Weekly Team discussions.



## Metrics:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Acccount ID | AUC | overall\_conversion | likely\_to\_win\_conversion | likely\_to\_win\_threshold |
| **21** | **0.733** | **53.49%** | **92.86%** | **0.811** |
| 5614 | 0.924 | 62.45% | 98.94% | 0.986 |
| 18349 | 0.832 | 27.87% | 83.33% | 0.684 |
| 33094 | 0.892 | 32.12% | 97.56% | 0.779 |
| 34715 | 0.851 | 8.09% | 39.04% | 0.221 |
| 38750 | 0.781 | 51.34% | 100.00% | 0.872 |
| 43908 | 0.834 | 28.54% | 77.78% | 0.592 |
| 53448 | 0.805 | 98.58% | 94.74% | 0.987 |
| 53997 | 0.944 | 7.95% | 55.00% | 0.259 |
| 55908 | 0.873 | 42.00% | 83.84% | 0.903 |
| 66489 | 0.884 | 18.56% | 72.73% | 0.544 |

## Product Mocks:

<https://freshworks.invisionapp.com/d/main#/projects/prototypes/16970766>

## Timelines:

1. **Beta readiness** - we are targeting this for the user conf - (ETA: April 20th)
2. **Scaling to 10 beta accounts** - (ETA: end of April)

## Action items for going live:

Engineering (FS):

1. Freeze the contract and give access to data through slave (ETA: This week)
2. UI for Deal insights (ETA: 2 weeks)

Engineering (ML):

1. Setting things up in Staging
2. Push scores via Sidekiq jobs
3. Production deployments
4. Automations

Data Science:

1. Make interpretability available for all TAGS (ETA: 2 weeks)
2. Deal insights and scores for all eligible customers (ETA: 2 weeks)
3. Lead insights for all eligible customers. (Also replace the current lead scoring) (ETA: 4 weeks)

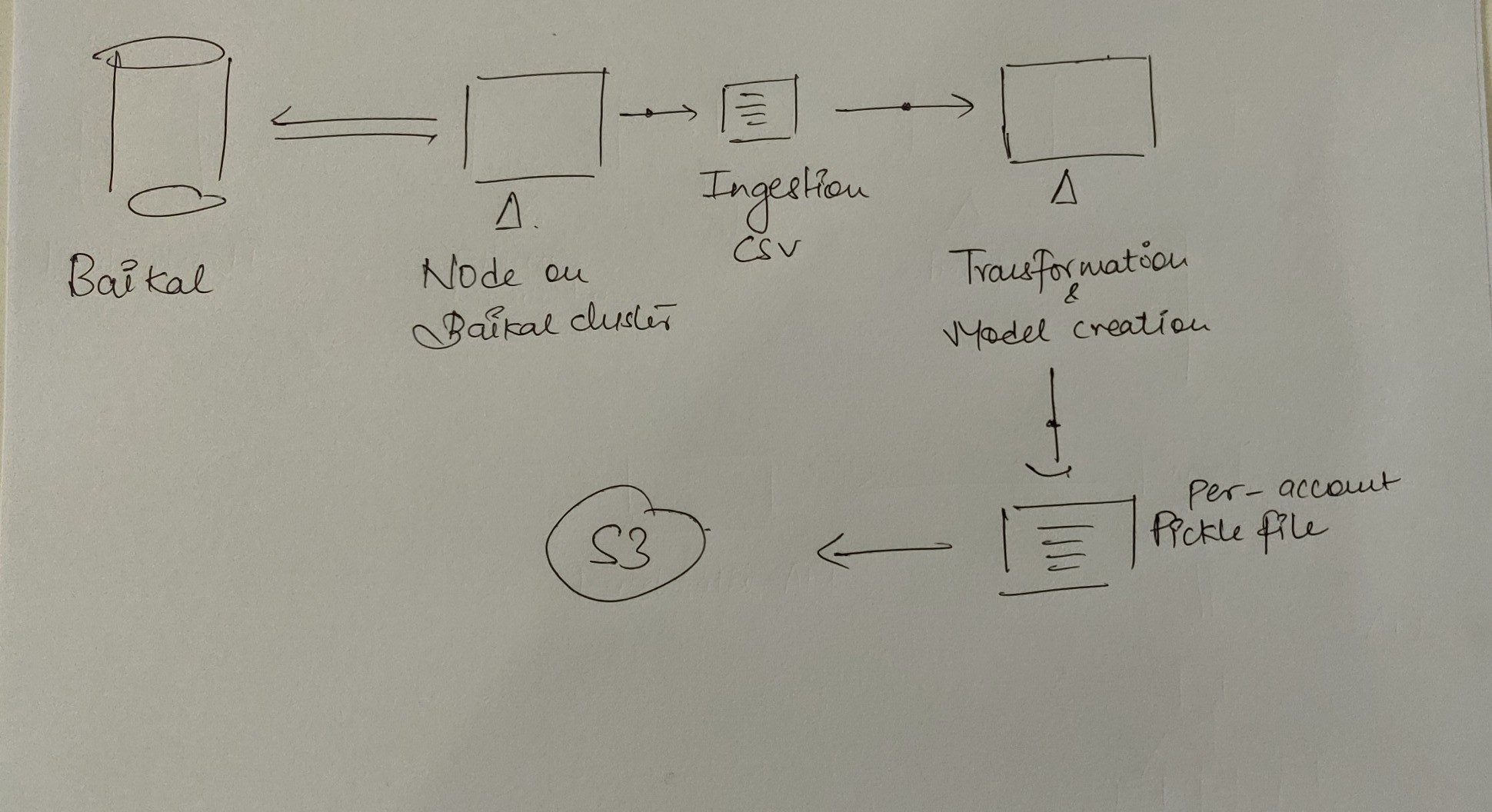
## Engineering Architecture:

The project will have two primary functions:

* Training
* Prediction

### Training

For training the model, we will use **Baikal** as the data source, since it has a *complete replica of the Freshsales database*. Being a data source at rest, it will give us the flexibility to make heavy Hive/Impala queries. As a result, the codebase will also be placed inside a new machine inside the Baikal cluster (only machines inside the cluster have access to Baikal data).



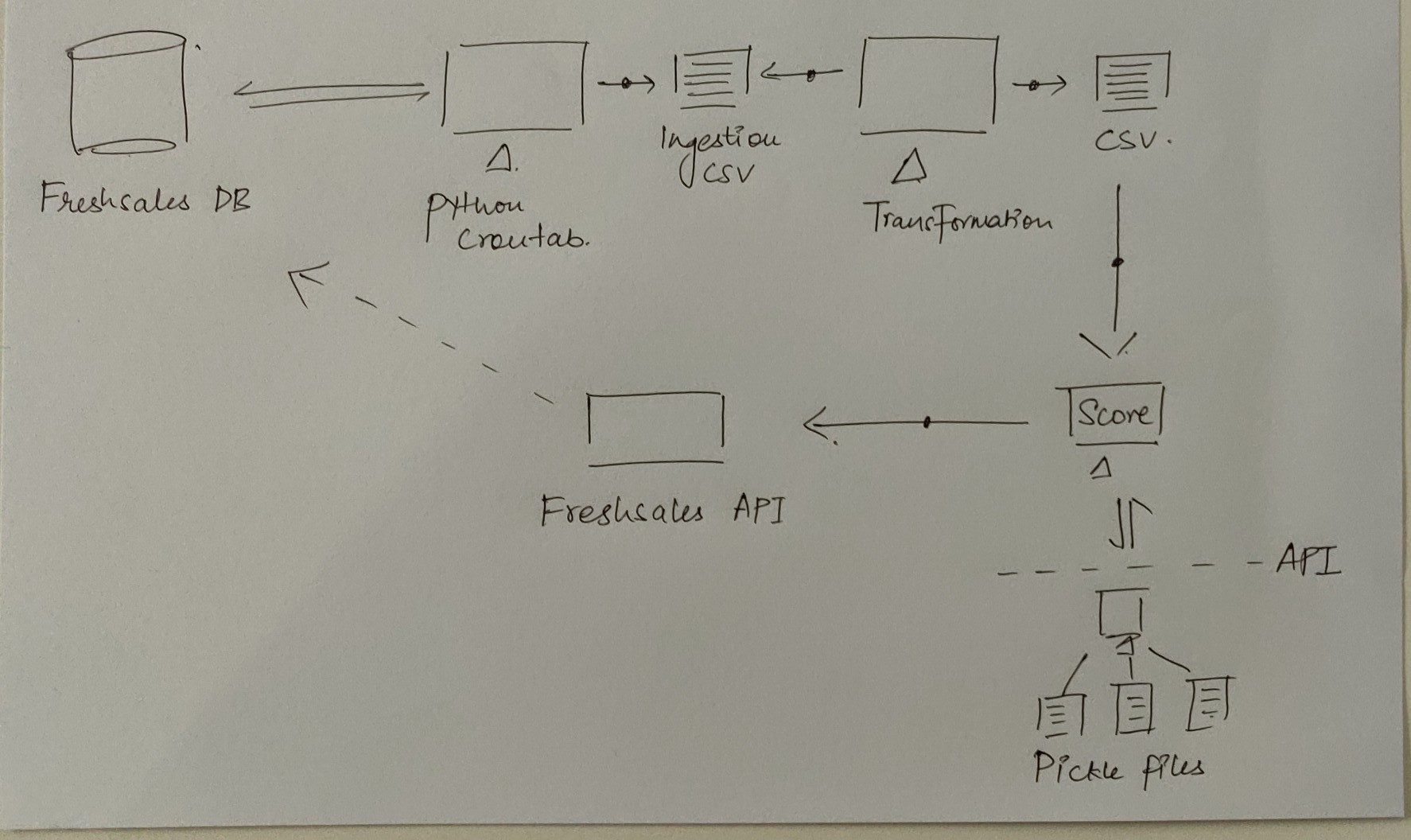
The code workflow will be designed to generate a CSV at every stage (ingestion, transformation, model training/creation), to have full visibility of data retrospectively. Every stage will also have a due monitoring process for health-checks (dots on the diagram). Barring the transformation and training code, the rest of the code will be written in Python.

The models will be stored as pickle files. On creation, these files will be uploaded to our S3 bucket.

*Note:* This section will be amended with the model training architecture upon further clarity there.

### Prediction

The prediction will run once every 30 minutes. The data source, due to the immediacy, will be the **Freshsales RDS**. The ingestion code will keep a log of the *last\_retrieved* timestamp.



The codebase will be placed inside the Freshsales stack itself. The flow will be similar to the training process, where every stage will generate a CSV, and have a set of monitoring health checks (dots on the diagram). For getting the score, the running cronjob will make a HTTP request to a lightweight microservice which will return with prediction results.

This microservice will be responsible only for storing the latest model pickle files and predicting results through them.