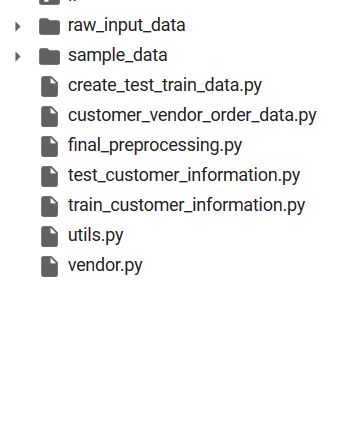
1. **Datafiles used in the Akeed competition :-**We used all the files provided in the competition homepage for creation of train and test data sets and treating the problem as a supervised learning (classification) algorithm.
2. Test\_location.csv
3. Orders.csv
4. Test\_customers.csv
5. Train\_customers.csv
6. Train\_locations.csv
7. Vendors.csv
8. SampleSubmission.csv

Usage of external datasets :- No external dataset was used in the formulation of the problem

**File Folder structure for input data creation and model training**



The input datasets mentioned above must be stored in a folder named – ‘raw\_input\_data’. All the modules described below must be stored in the same location as the folder- ‘raw\_input\_data’.

There will be a separate notebook where the model will be trained and output attained.

1. **Code structure**

**Step 1:- Dataset creation and feature Engineering**   
  
**(Base Module) utils.py :-** This module contains all the utility functions to be used throughout the code. The functions include:-

1. Distance based function calculation (like functions for haversine, Manhattan or Bearing distance calculations),
2. Latitude and longitude based location mapping functions (to decide whether a geological point is in the North-East or South West after we see the sign of the latitude and longitudes given)
3. Vendor time based functions like deciding the total functional hours of a restaurant, whether a restaurant provides dinner, lunch or breakfast based on the operational timings.
4. Some helping functions of memory reduce, groupby and joins to be used repeatedly in the course of the preparation of train and test data.

**train\_customer\_information.py:-** This module takes train location and train customer location as input, joins both the dataframe based on the location coordinates and then creates many latitude and longitude based features. To make sure the segment runs, one has to update the train\_locations\_path and train\_customer\_path variable.

We did create a lot of features based on the aggregations done on latitude and longitude and their transforms. This was done after we followed the discussion in the link :- <https://zindi.africa/competitions/akeed-restaurant-recommendation-challenge/discussions/1882> . We were using standard mathematical distance measures first. But the aggregations gave a boost to the scores and we realised that if we create more features from the obfuscated /masked coordinates, we were able to generate some reasonable interaction features that can perform reasonably well and can aid the model when we do clustering in the latter part   
Dataset path to be changes :- path of train\_locations.csv and train\_customers.csv

**test\_customer\_information.py :-** This module does the exact same process as the train\_customer\_information.py but for the test locations and test customers.  
Dataset path to be changes :- path of test\_locations.csv and test\_customers.csv

**vendor.py :-** This module takes care of the preprocessing of vendor data (to be joined later with both train and test customer information created in the previous steps). The path having orders.csv and vendors.csv needs to be specified in the variable – order\_location and vendor\_location for the code to run.  
In vendor.py, we perform the following functions :-   
a. Load order and vendor dataset  
b. Time based vendor features (like what are the shift timings?, Is nigh service available? What type of food is served based on timings(dinner, lunch etc) ?   
c. Primary tags based sparse matrix creation using Count Vectorization  
d. Aggregate feature creation using orders served previously to quantify vendor’s capabilities in terms of likeliness, preparation time, maximum orders served, average distance covered etc)   
The final dataframe created will be vendor\_df.  
Dataset path to be changed :- Path of order.csv and vendor.csv

**Customer\_vendor\_order\_data.py :-** This module merely captures the interaction between the customer and vendor latitude and longitudes. It takes into account the vendor\_df and order\_df from previous step and train\_customer\_information from the first step

**Create\_train\_test\_data.py :-** This module creates the test and train data based on the customer information and the vendor information captured till now. While the data is being created, we also take into account the customer latitude and longitude value based rotational transformations.

**Final\_preprocessing.py:-**   
This module was meant to perform unsupervised learning on both train and test datasets. We form clusters here based on individual customer as vendor locations as well as the data having customer-vendor location pairs.  
We also calculate the PCA based on the latitude and longitude pairs for both customers and vendors to capture the variation. Finally we use online dictionary learning to capture the most variable inducing component in the sparse matrix created in vendor.py on food types.

**Final\_model.ipynb :-**  This module eventually creates the train and test data and performs the training on it. The first step in model building process was to do label encoding for the categorical columns developed in the feature engineering process.

We used LighGBM model only for the training purpose. Further we used 3 lightgbm models and did voting classification of them. The 3 lightgbms selected were performing really good individually and with voting classifier, we got even better results. Finally a threshold of 0.57 was set to get the best result attainable.

**Infrastructure used:-**

Since the data preparation step is an infra heavy and a time consuming one, we made sure that we modularise it to make error tracking easy and keep data preparation separate from model training to reduce burden on RAM. We used Google Colab Pro extensively for it(25GB RAM) due to absence of local infrastructure to run and preprocess the datasets

Bottlenecks (time consuming blocks in the process):-

1. Clustering for customer-vendor combination of latitude and longitudes (approximately 10-15mins)
2. Model.fit() :- 2 hours to train the data
3. Wherever possible, we used reduce\_mem\_usage() function to accommodate the data in the RAM provided in the colab. This may take a little bit of time at times to reduce the memory

|  |  |  |
| --- | --- | --- |
| **Libraries** | **Version** | **Purpose** |
| Pandas | 1.0.5 | Data read and preprocessing |
| Numpy | 1.18.5 | Mathematical functions |
| Lightgbm | 2.2.3 | Model training |
| Sklearn | 0.22.2.post1 | unsupervised learning and model training |

**Structure for train and test data creation and feature engineering**

utils.py

Final\_preprocessing.py

Create\_test\_train\_data.py

Customer\_vendor\_order\_data.py

Vendor.py

test\_customer\_information.py

Train\_customer\_information.py