Innovation Partnership Collaboration between NCI and Glantus Ltd.

Executive Summary

The NCI-Glantus IPP project was funded under the Enterprise Ireland Innovation Partnership Programme, which is co-funded by the European Regional Development Fund. The project got underway in August 2019. Glantus - an innovative developer of business led data processing and analysis platforms - approached NCI to examine the possibility of improving Glantus' current platform with the inclusion of specific analytical tools. The aim of the project was to carry out research using a retail Machine Learning environment to build new advanced models such as product prediction, customer segmentation and voucher recommendation for the Business Intelligence market. The project team consists of Dr Anu Sahni (Principal Investigator), Dr Pramod Pathak (Co- Principal Investigator), Dr Paul Stynes (Project Manager), Dr Cian O'Hara (Post-Doctoral Researcher), Pramit Kumar (Technical Lead), Arjit Agrawal (Research Assistant), Abhishek Jain (Research Assistant), Dr James Little (Senior Data Scientist), Nilay Kumar (Consultant), and Manus McDonald (Consultant). The team has fully analysed and implemented machine learning in retail to build an end to end cloud Market Basket Analysis (MBA) model and API delivered through Azure with User Interface to recommend products. The end to end on-premises MBA model was also developed with API delivered through Flask, Docker and Kubernates.

The research was further extended as a research project with a master student, Samruddhi Kanhere, to investigate if the clustering-based approach can reach a similar solution even after reducing the dataset size.

The research paper was orally presented at the 28th FRUCT conference, https://www.youtube.com/watch?v=odW6b9zWSlw. It was also published in the Full Papers

section of the conference proceedings, https://fruct.org/publications/fruct28/files/Kan.pdf, and sent to IEEE Xplore library, Scopus and many other indexes, as specified at www.fruct.org/cfp.

The authors acknowledge the support of Glantus (www.glantus.com) in the research paper and sincerely thank them for providing the real world data for this research as a part of an Enterprise Ireland funded Innovation Partnership Programme.

Testimonial from Glantus:

"Deriving Value from Data & Analytics is at the heart of the Glantus proposition to transform the Finance function. To deliver the most advanced technology solutions to our customer base across the world, we need to continuously grow and innovate our offering. To achieve this, takes a lot of research and development but also requires collaboration with leading organisations across industry and academia.

Our collaboration with NCI helps us to shape our approach to solving the key business problems that our customers face, by providing meaningful outcomes and insights that matter. We would encourage other organisations to engage in this style of innovation partnership as a means to strengthen the relationship between academia and industry."

Joe Keating
Chief Data Officer

Geoff Keating
Chief Technology Officer

Introduction

The over-arching objective of this project was to develop pre-commercial algorithms to provide product recommendations to vendors. The main results can be broken into two parts. The successful implementation of a market basket analysis recommender system which has been integrated into Glantus' in house infrastructure, and the creation of a

machine learning framework using Databricks MLFlow to streamline current and future ML projects. To achieve this the following sub-objectives were identified:

1. Examine and clean all incoming data streams.
2. Build a recommender system using approaches such as association rules.
3. Develop a library of appropriate machine learning techniques to model such demand data.
4. Design an architecture that can be integrated with current products of Glantus.

Market Basket Analysis

Methodology

This section defines all requirements for the potential factors of input to Market Basket Analysis - Apriori Algorithm and FP Growth. It details data input, processing, transformation, preparation, and delivery requirements.

Product IDs were replaced by product names to make results more interpretable. Data was converted from CSV to the scientific notations format and finally into numeric format.

PROMOPAY, a UK based company that provides software for till and voucher to retail stores, extracted PMRB file contained millions of transactions but only 25000 rows were selected due to processing limitations.

Stages:

- Pre-processing code
- Extracting and looking up all the item IDs in the basket of each transaction (problem specific)
- Model Generation API
- List of lists for MBA algorithm
- Fixed confidence limit
- Generation of ruleset
- Application of Model API
- Recommendations of items to a basket using a fixed heuristic (recommend on pairs) approach based on ruleset
- MBA used to co-locate items in the store or suggest marketing strategies

- Product category ruleset wasn't implemented although an extension to the algorithm was developed

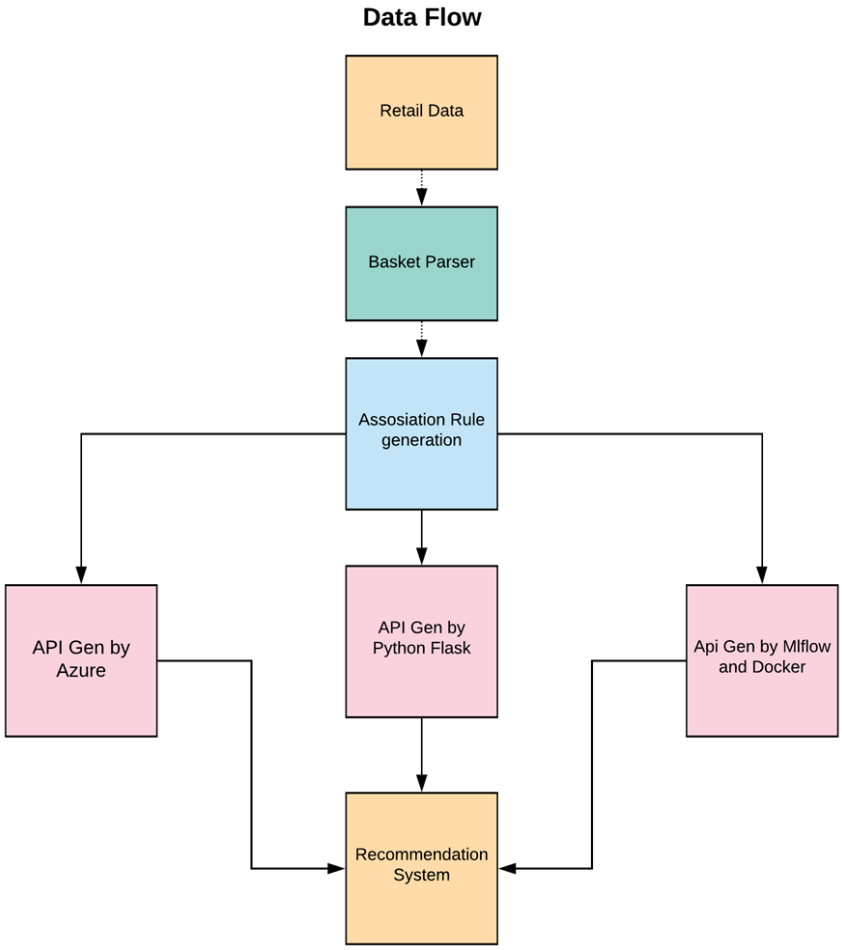


Figure1: Workflow

Note: Retail product recommendation is traditionally based on similarity measures between customers or between products. This was not considered as the customer data and detailed product data was not provided.

Recommendation Logic:

- Master Files contained all the rules generated using the Parallel FP-Growth (PFP) algorithm.
- Creation of the master file took place during the training phase

An example of recommended products with Support >= 35% by removing duplication and giving higher priority to the items with offers.

Product 1	Product 2	Actual Recommeded Product	Support	Confidence
P1	P2	**P11**	0.9	0.8
P2	P3	**P15**	0.4	0.45
P3	P4	**P12**	0.8	0.7
P4	P1	**P16**	0.35	0.4

MBA Implementation

Daily transactional data which was in .txt format was imported using the Panda DataFrame library. Each customer transaction, the list of items, for each day (3 million transactions) was extracted. This list of items was used for mining different association rules. The generated rules were used in the Python Flask application to get the customer's purchased items and return the recommended products on the basis of the strength of rules. The strength of a rule is calculated by the support and confidence values.

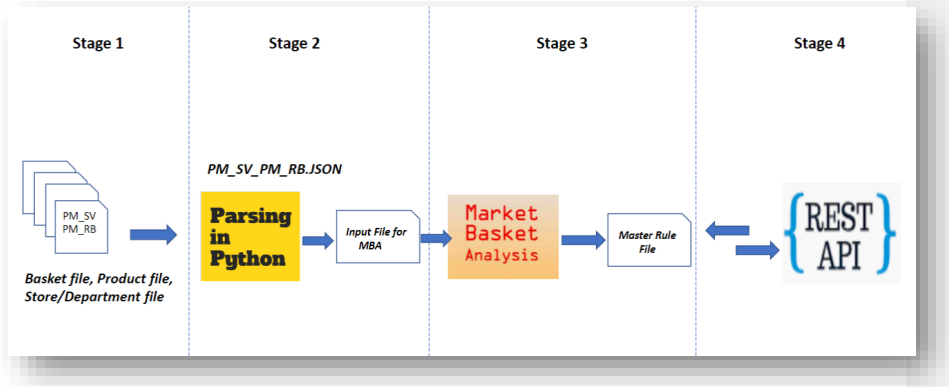


Figure 2: Various stages of development cycle

The transactional files where products were represented by their barcodes and not names as barcodes are unique. With the help of metadata files, each barcode was later mapped back to extract their names. The first two rows of rule file are shown below:

FirstItem	SecondItem	TargetItem	Support	Confidence

6100000000193	5031390114669	9770307268922	0.0007	0.2264
5031390114553	5000157062680	6100000000193	0.0007	0.1449

Data Cleaning and Transformation

JSON column from PMRB file were converted into Transactional format using R and Excel.

Data Specifications

25000 Rows were selected randomly out of 2 million rows due to system processing constraints. The Product ID did not match with Products given in the XML.

Hence, substitute product names were used in this analysis.

- Out of 9000 unique products discovered in the 25000 transactions, 1300 product details were found in the given dataset.

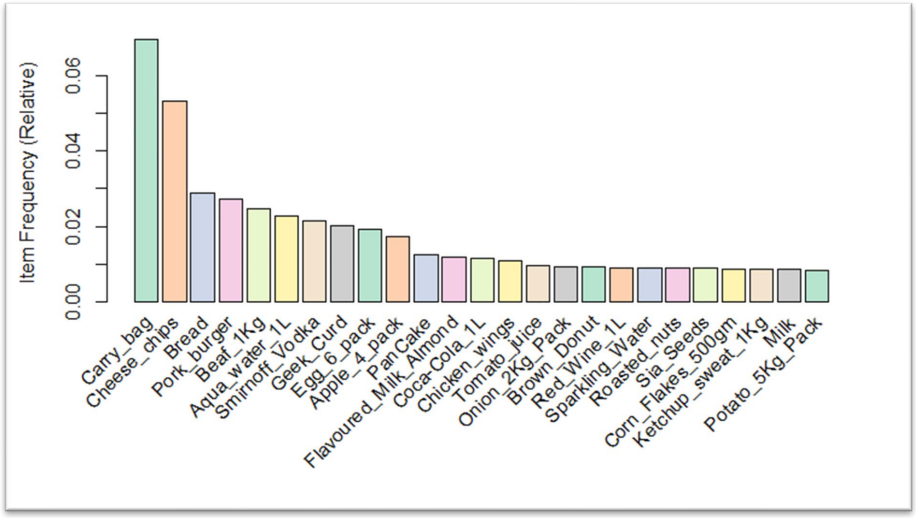


Figure 3: Product sale

Commented [AS1]: Pramit – explain the figure

For example, on 01-12-2018 Promopay application generated a transactional file 20181202_PMRB_0001.CSV. The file had many columns with product transaction details in

JSON format. Each transaction consisted of product ID (mainly barcode), price, and quanity. In the initial stage, the information was extracted into more readable tabular format using JSON parsing to tranpose rows into columns. Then each barcode was matched with product metadata as shown in the figure below.

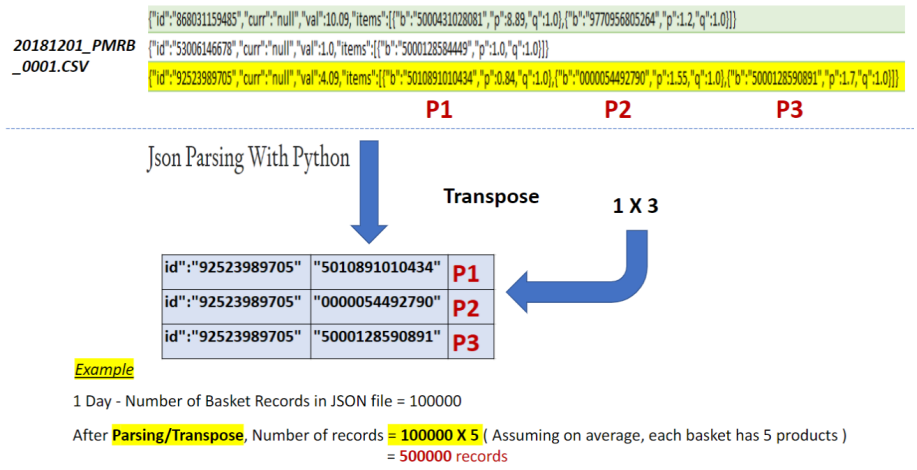


Figure 4: JSON to tabular format

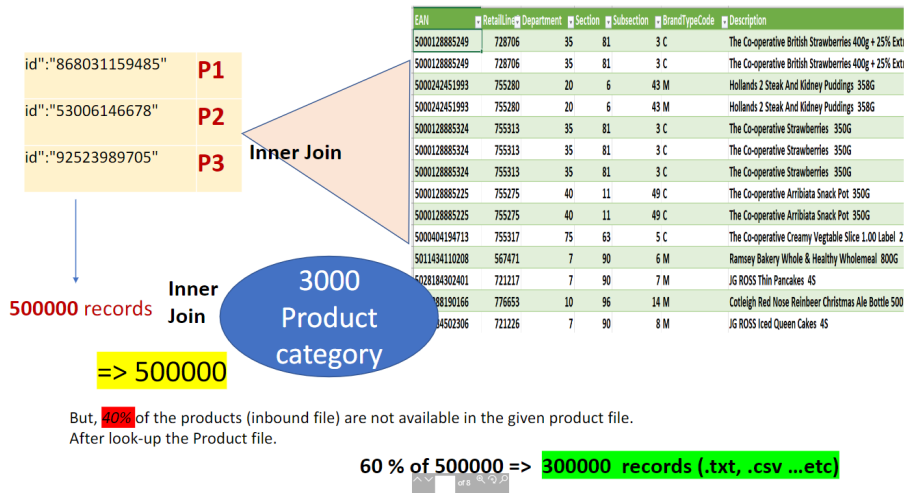


Figure 5: Tabular format mapped with product metadata

After the data cleaning and preparation stage, the data was fed into the machine leaning model (MBA model), which produced the set of rules in a separate file.

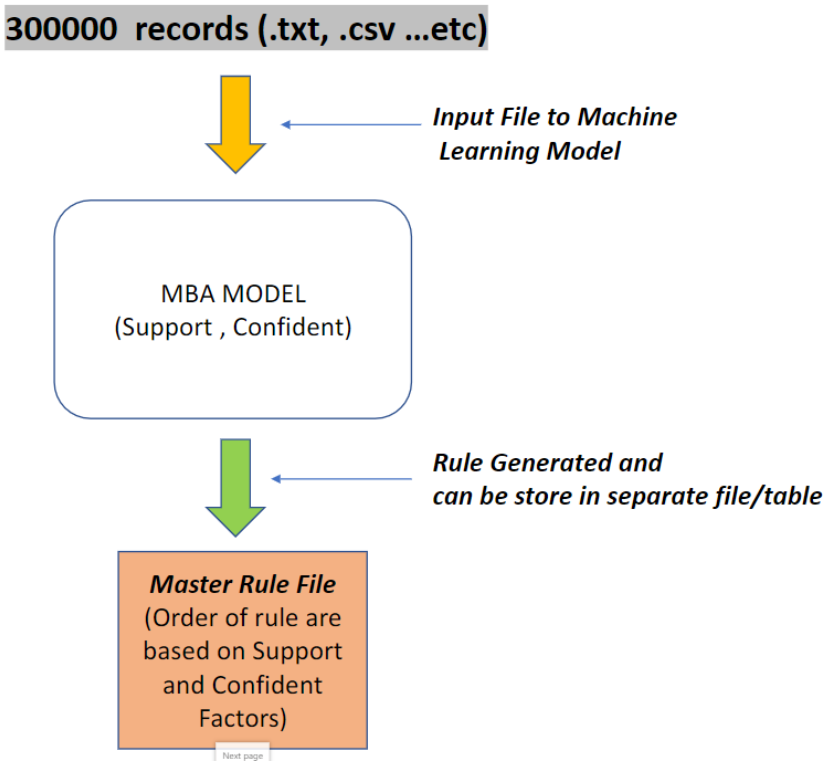


Figure 6: Rules file generation

One of the requirements was that the company internal business portal should allow its users to select minimum 2 products as an input to get a recommendation system. In order to achieve this, API signature were created, which gets the user input to the recommendation system and outputs recommended products.

API details and standards

- API signature [Development Environment]

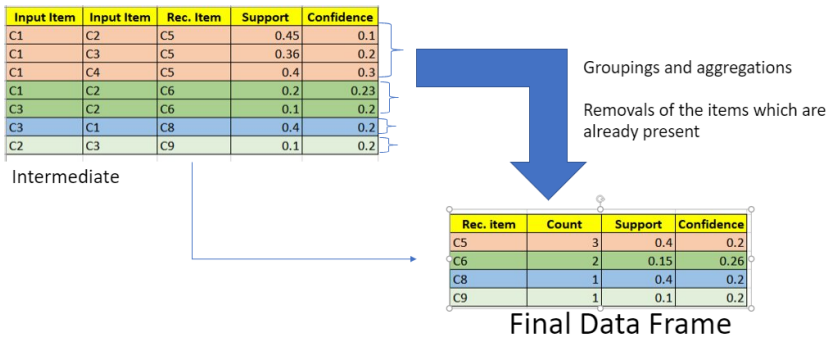
http://127.0.0.1:5000/api/items

- API Payload – Body (sample)

```
{
  "items": [
```

```
    "0000000007535",
    "0000054491472",
    "7622210286918"
  ]
}
```


API Working Logic Overview:



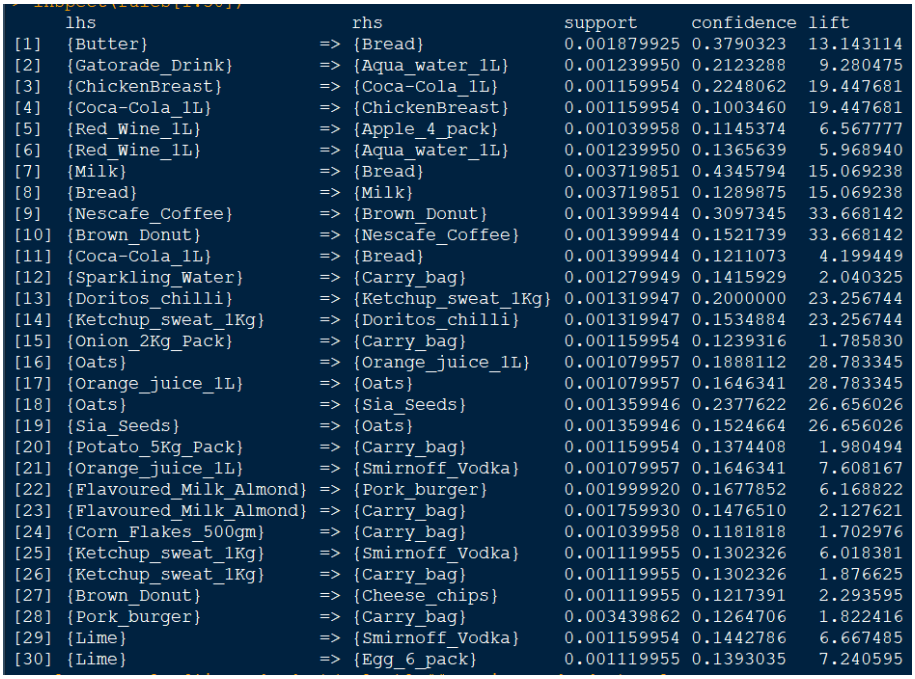
API Features

- Minimum latency ~100 milliseconds.
- Refactored and functionalized code.
- Adequate Exception Handling.
- Plug and use files.
- No time taking SQL connections, only data frames used
- ~ 21 rules (both type of rules) present from 10K item baskets
- Only make predictions for more than 2 products.

Use case 1

Parameters for rules generation

Lift, support, and confidence were used for the rules evaluation. Rules were also pruned to remove any duplicity.
Confidence: 10%
Support: 0.1%
Min Rules length: 2



	lhs		rhs	support	confidence	lift
[1]	{Butter}	=>	{Bread}	0.001879925	0.3790323	13.143114
[2]	{Gatorade_Drink}	=>	{Aqua_water_1L}	0.001239950	0.2123288	9.280475
[3]	{ChickenBreast}	=>	{Coca-Cola_1L}	0.001159954	0.2248062	19.447681
[4]	{Coca-Cola_1L}	=>	{ChickenBreast}	0.001159954	0.1003460	19.447681
[5]	{Red_Wine_1L}	=>	{Apple_4_pack}	0.001039958	0.1145374	6.567777
[6]	{Red_Wine_1L}	=>	{Aqua_water_1L}	0.001239950	0.1365639	5.968940
[7]	{Milk}	=>	{Bread}	0.003719851	0.4345794	15.069238
[8]	{Bread}	=>	{Milk}	0.003719851	0.1289875	15.069238
[9]	{Nescafe_Coffee}	=>	{Brown_Donut}	0.001399944	0.3097345	33.668142
[10]	{Brown_Donut}	=>	{Nescafe_Coffee}	0.001399944	0.1521739	33.668142
[11]	{Coca-Cola_1L}	=>	{Bread}	0.001399944	0.1211073	4.199449
[12]	{Sparkling_Water}	=>	{Carry_bag}	0.001279949	0.1415929	2.040325
[13]	{Doritos_chilli}	=>	{Ketchup_sweat_1Kg}	0.001319947	0.2000000	23.256744
[14]	{Ketchup_sweat_1Kg}	=>	{Doritos_chilli}	0.001319947	0.1534884	23.256744
[15]	{Onion_2Kg_Pack}	=>	{Carry_bag}	0.001159954	0.1239316	1.785830
[16]	{Oats}	=>	{Orange_juice_1L}	0.001079957	0.1888112	28.783345
[17]	{Orange_juice_1L}	=>	{Oats}	0.001079957	0.1646341	28.783345
[18]	{Oats}	=>	{Sia_Seeds}	0.001359946	0.2377622	26.656026
[19]	{Sia_Seeds}	=>	{Oats}	0.001359946	0.1524664	26.656026
[20]	{Potato_5Kg_Pack}	=>	{Carry_bag}	0.001159954	0.1374408	1.980494
[21]	{Orange_juice_1L}	=>	{Smirnoff_Vodka}	0.001079957	0.1646341	7.608167
[22]	{Flavoured_Milk_Almond}	=>	{Pork_burger}	0.001999920	0.1677852	6.168822
[23]	{Flavoured_Milk_Almond}	=>	{Carry_bag}	0.001759930	0.1476510	2.127621
[24]	{Corn_Flakes_500gm}	=>	{Carry_bag}	0.001039958	0.1181818	1.702976
[25]	{Ketchup_sweat_1Kg}	=>	{Smirnoff_Vodka}	0.001119955	0.1302326	6.018381
[26]	{Ketchup_sweat_1Kg}	=>	{Carry_bag}	0.001119955	0.1302326	1.876625
[27]	{Brown_Donut}	=>	{Cheese_chips}	0.001119955	0.1217391	2.293595
[28]	{Pork_burger}	=>	{Carry_bag}	0.003439862	0.1264706	1.822416
[29]	{Lime}	=>	{Smirnoff_Vodka}	0.001159954	0.1442786	6.667485
[30]	{Lime}	=>	{Egg_6_pack}	0.001119955	0.1393035	7.240595

Table 1: Rules

Presentation of Product Links using Graph.

The bigger the circle, the stronger the rule.

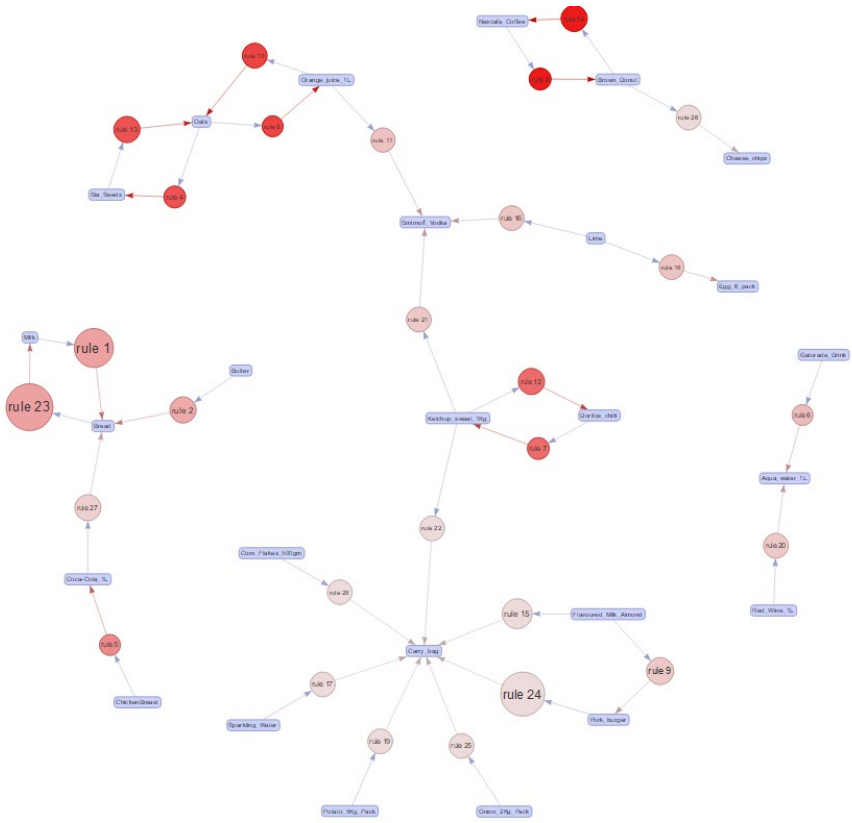


Figure 7: MBA Link Graph

Another Option for link graph

Bigger circle means more support. Links show relationship between the products.

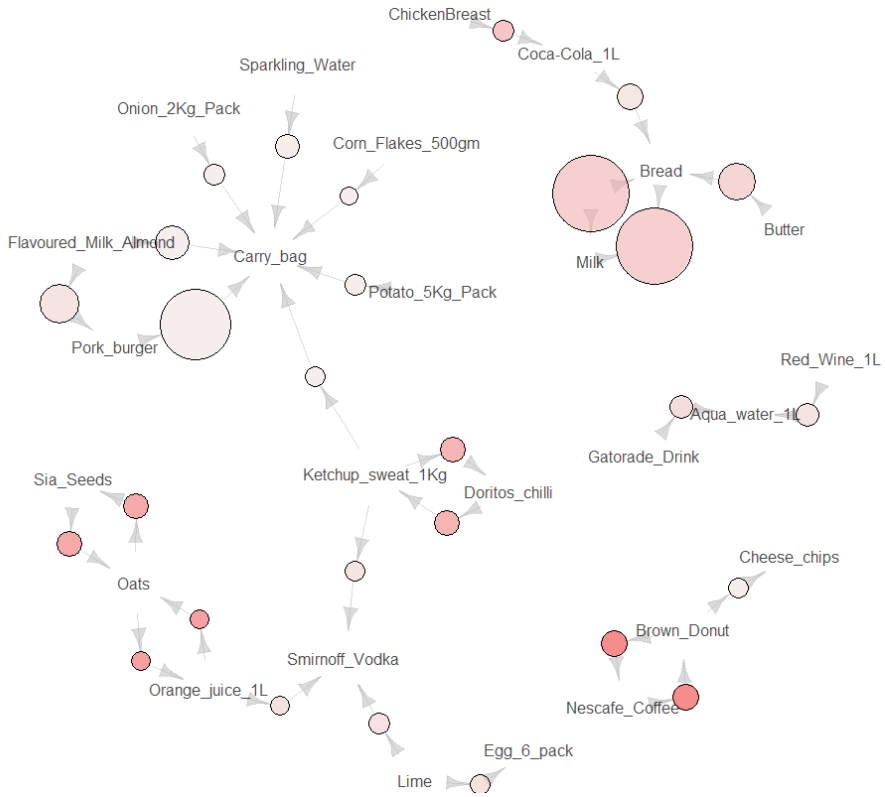


Figure 8: MBA Link Graph

Individual Product Association Graph

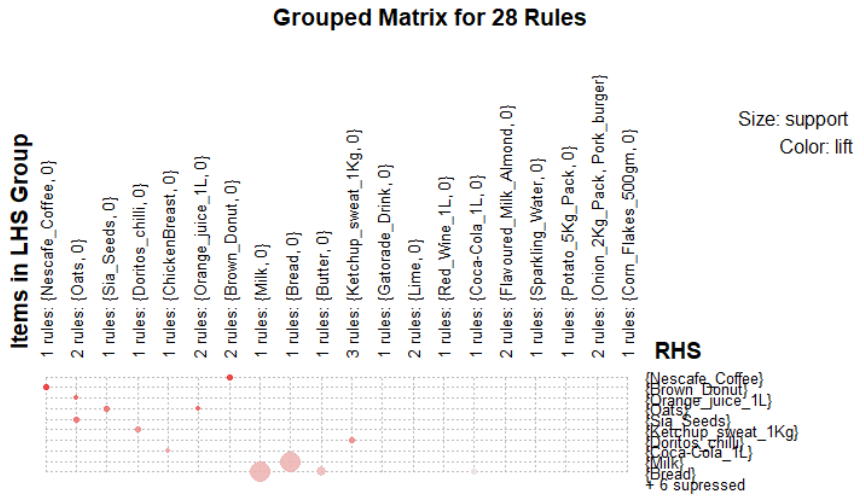


Figure 9: Association between individual products

Use case 2

Parameters for rules generation

Lift, support, and confidence were used for the rules evaluation. Rules were also pruned to remove any duplicity.

Confidence: 50%

Support: 0.04%

Min Rules length: 3

	lhs		rhs	support	confidence	lift
[1]	{Mix_veg_salad,Speghetti}	=>	{Cheese_granuels}	0.0004399824	1.0000000	531.93617
[2]	{Cheese_granuels,Speghetti}	=>	{Mix_veg_salad}	0.0004399824	1.0000000	925.96296
[3]	{Buns,Hot_chilli_sauce}	=>	{Sausage}	0.0004399824	0.8461538	846.18769
[4]	{Buns,Hot_chilli_sauce}	=>	{Mix_veg_salad}	0.0005199792	1.0000000	925.96296
[5]	{Buns,Cheese_granuels}	=>	{Mix_veg_salad}	0.0005199792	1.0000000	925.96296
[6]	{Honey,Musli_fruit&nut}	=>	{Crisp_Wafers}	0.0004399824	0.7333333	273.64279
[7]	{Crisp_Wafers,Musli_fruit&nut}	=>	{Honey}	0.0004399824	0.5238095	256.77965
[8]	{Butter,Milk}	=>	{Bread}	0.0004399824	0.5238095	18.16333
[9]	{Hot_chilli_sauce,Mix_veg_salad}	=>	{Cheese_granuels}	0.0005599776	0.8750000	465.44415

MBA on Azure ML Studio

Market basket analysis machine learning model was deployed on Microsoft Azure. The files **"Migration.zip"** and **"Migration input.csv"** are needed to recreate the example given below.

There are several steps to follow from the initial data set to building and deploying a machine learning model using Azure.

Figure 7 shows an example of a working experiment, which should run with the attached files.

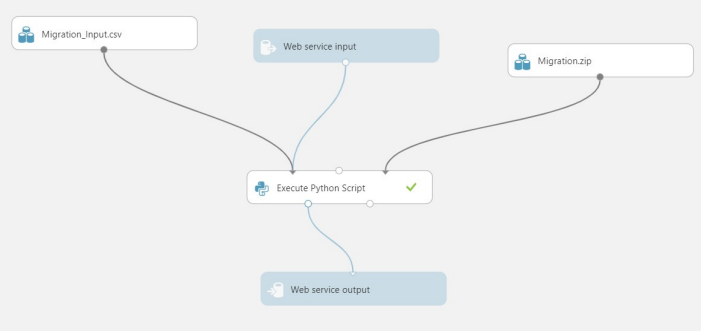


Figure 10: A simple Azure experiment.

The experiment contains a number of separate blocks joined together and described as follows. Each will be explained in more detail later.

1. **Execution of Python Script:** This is the most important block. This is where the custom Python code is entered, in this case the frequent pattern growth algorithm for market basket analysis. The code can be found in **"Migration.zip"**, called **"working azure code.txt"**.

   In this example, we filter out the data according to the parameters: PriceB=[1,2,3], RegionN=["North"], StoreFormat=["Food"], MembershipR=["Durham"]. These parameters were chosen to reduce the amount of data as it takes time to run the analysis. We can change these parameters to increase the volume of data set. For example, here we are considering only one area and region. But we can always add more regions and areas to increase the data.

2. **Migration input.csv:** This data set is used to specify the format of the web service to input data. Each column of this data set corresponds to one input field for the Azure

input interface, so by adding a column we can add an additional input parameter. In accordance with the Azure instruction we need put some default values in these columns so that Azure can interpret the type of data it is going to process from user input. The example used here, shown in Table 1, creates four input fields.

PriceBracket	RegionName	input item1	input item2
2	NORTH	BREAD WHITE Standard 800G	Cider STD BTLE DRY

Table 2: Migration input.csv

3. **Migration.zip:** This zip file contains all the required data sets and Python software libraries, which are needed for the Python script to run, as well as the script itself. In this example these files are as follows:

Name of File	Importance
Processed _Product.txt	Details regarding products
CoopStore Attributes.csv	Used to filter out the data from BigTempRB.txt on the basis of Price band, Region and StorID.
BigTempRb.txt	Contains item JSON strings. Merged Processed product table ("EAN") with "ItemID"
pyfpgrowth (folder)	Contains Python required packages

Table 3: Contents of Migration.zip

4. **Web service Input/Output:** These blocks create the input and output interfaces needed to run the experiment. The input block coupled with the Migration input.csv block will create an interface with the appropriate number of data fields to be fed into the python code block. The output block will then return a recommendation found from the market basket analysis.

Steps to deploy the model and get recommendation:

1. Creating Azure account: To create this model, an Azure cloud services account should be created at https://studio.azureml.net. Uploading the data sets

2. Import data sets into Azure. On the left side of the dashboard there is a data set tab. From here one can add new data sets by clicking new and upload to Azure from the local device.

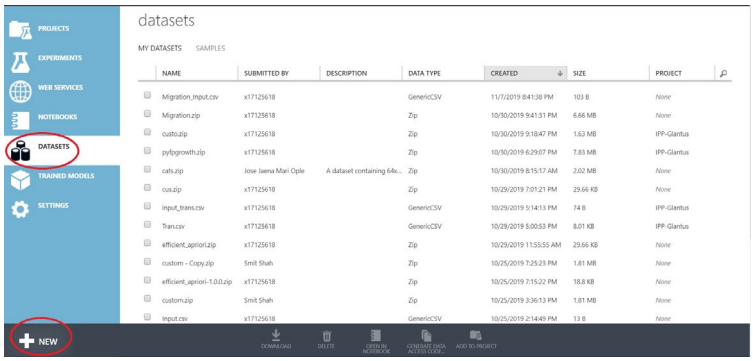


Figure 11: Uploading data sets to Azure.

3. Start a new blank experiment from the Experiments section on the left hand side.
4. Drag and drop the imported data into the experiment, as shown in Figure 12.
5. . The saved data sets are displayed in the left panel under Saved Datasets in the My Datasets option.

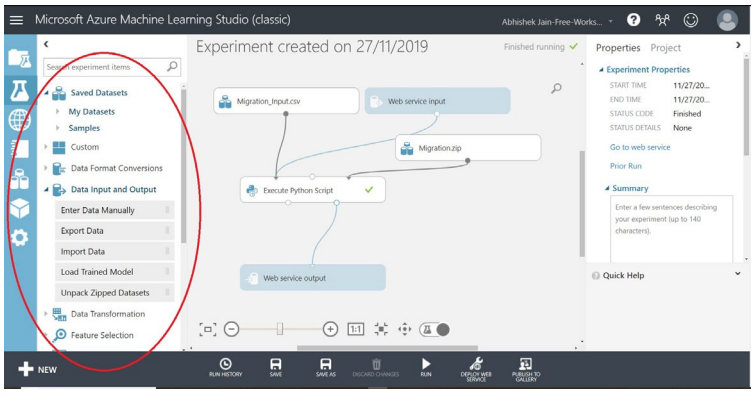


Figure 12: Adding data sets and python code blocks to an experiment

6. The next step is to add Python script to create the Machine Learning model. First, import the “Execute Python Script” module from the left that allows to write code

and join the imported datasets into the coded environment. There can be three inputs in the python script module, Figure 13 below.

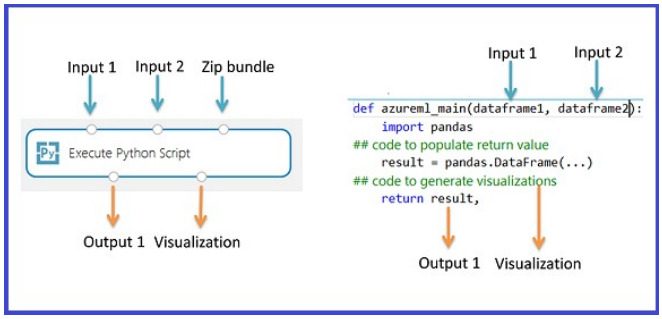


Figure 13: Schematic of the "Execute Python Script" box

7. By default, this module takes Input 1 and Input 2 in a DataFrame format. To access additional data sets or files which are not part of the dataset, but are essential for running the code like Python libraries, can use the third option, "Zip bundle". The 'Zip bundle' option takes data files in zip format by default.
8. There are some instructions from the Microsoft website which explains the working environment of the Azure Python module. The Python script text box is pre-populated with some instructions in comments, and sample code for data access and output. This code was replaced with the contents of **"working azure code.txt"**. The zip file containing **mymodule.py** was imported it by using **import mymodule** in the python code.
9. Next, the "Import Web service" and "Export Web service" boxes were added. These can be simply dragged from the web service tab on the left of the dashboard or added automatically by selecting the Web Services Deploy button as shown in the figure. If adding automatically using the deploy button, the web service input box correctly detects the input 1 of "Execute Python Script". Coupled with the "Migration input.csv" block creating the required input interface. There are two output ports in Execute Python Script. The first port is for output and the second one is for visualisation. Unfortunately, when using Deploy Web Services, the "web service output" was connected to the wrong port. We had to manually drag the connection to the output port of "Execute python script".

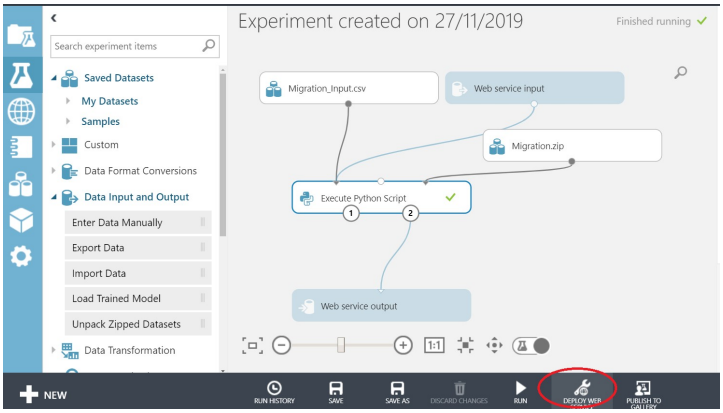


Figure 14: Adding web service input and output

10. The first step to get recommendation after the experiment completes, is to execute the model by clicking the run button at the bottom of the dashboard. If it has been created properly and no errors are encountered while running the python script, the run should exit successfully. It is at this stage that the frequent pattern growth algorithm mines the dataset for associations between items and builds the desired recommendations. Any errors, such as missing data or Python packages, will show up at this stage.

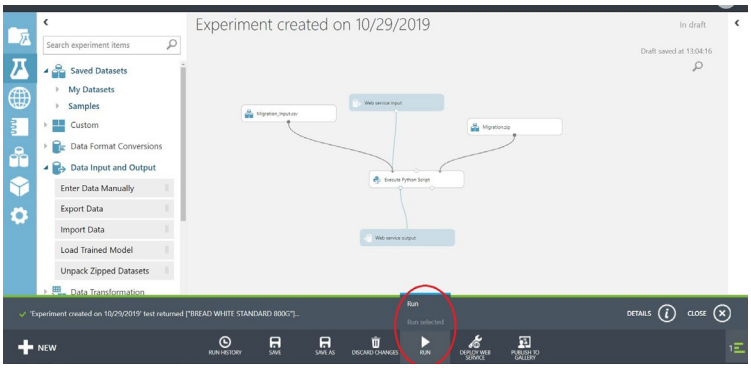


Figure 15: Running the model

11. To test the recommendation engine, press the Deploy Web Service button at the bottom.

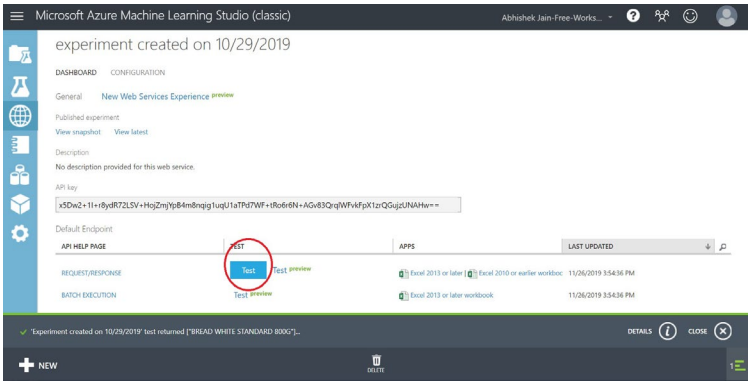


Figure 16: Testing the model

12. After pressing the deploy button, you will be asked if you want to overwrite the previously implemented experiments, select yes to navigate to a new screen for testing.

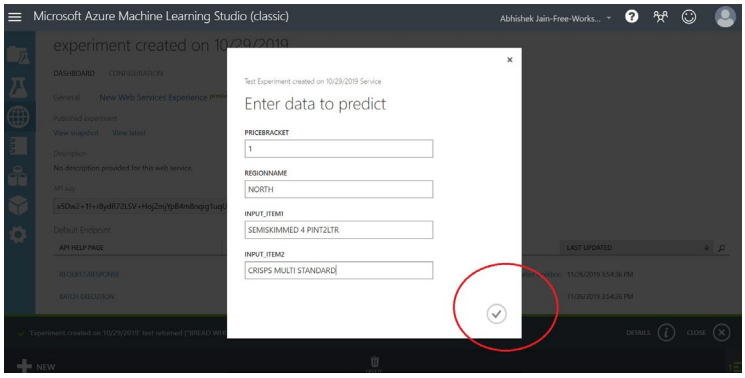


Figure 17: Recommender system input interface

As an example to make recommendation, the following values were entered, Figure 8.

- PriceBracket: 1
- RegionName: NORTH

- InputItem1: SEMISKIMMED 4 PINT2LTR

- Input_Item2: CRISPS MULTI STANDARD

White bread was recommended as shown in Figure 9.

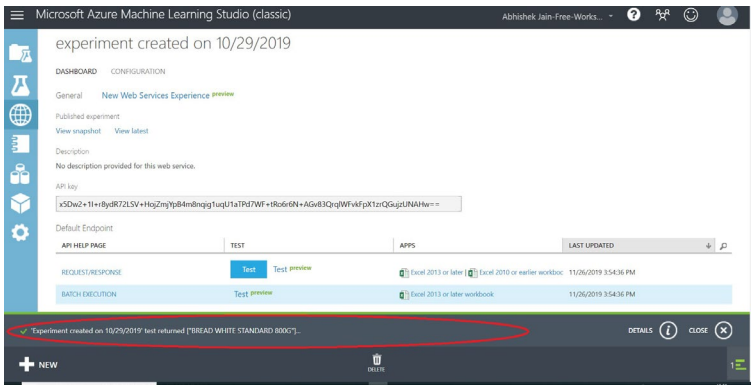


Figure 18: Product recommendation

13. Finally, to create an API from Azure, click on Web Services Experience

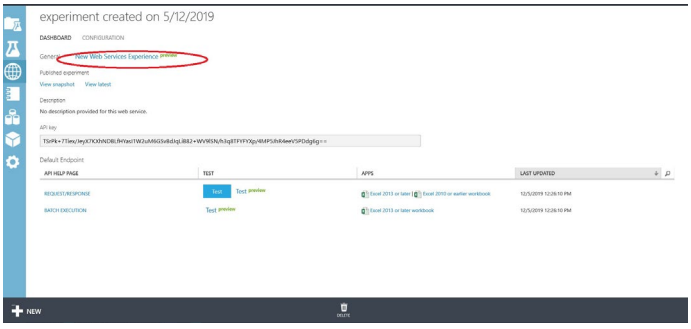


Figure 19: Selecting web services experience

14. Click on the “Consume” option to select the software language for which you want to create the API. We selected Python version 3.

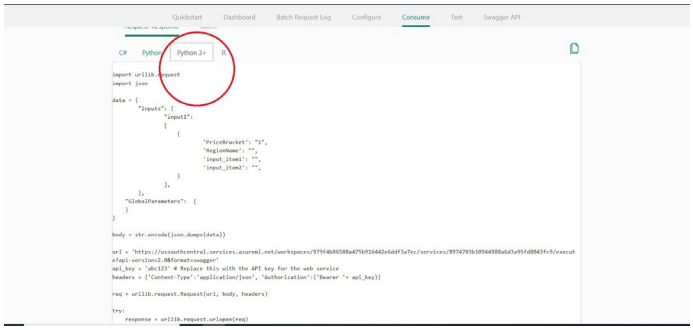


Figure 20: API code

On-Premises Flask/Container Market Basket Analysis

We built an API using the Python Flask framework to use its WSGI (Web Server Gateway Interface) functionality. This API can be used to send data and receive results in JSON format. An example of the JSON request that is passed in the product list representing a basket is shown below:

```
http://127.0.0.1:5000/api/v1.0/items?itemList={""id"":""1234"",""curr"":""null"",""val"":14.6,""items"":[{""
b"":""6100000000193"",""p"":5.0,""q"":1.0},{""b"":""6100000000193"",""p"":5.0,""q"":1.0}]}
```

The above **itemList** variable stores the product EAN Barcode, represented as key “b”. This is sent to the server where the Flask application uses the prebuilt rule file to get the recommendations for the whole basket based on the individual and pairs of products therein. Only the strongest, different product recommendations are made.

The output of the above request is as follows.


```
[
  {
    "Product Name": "EUROMILLIONS",
    "Product Category": "NATIONAL LOTTERY PRIZES",
    "RecommendedItem": "2083803000000",
    "CountItem": 1,
    "Support": 0.002,
    "Confidence": 0.1754
  },
  {
    "Product Name": "Not Available",
    "Product Category": "Not Available",
    "RecommendedItem": "5031390114669",
    "CountItem": 1,
    "Support": 0.001,
```

```
    "Confidence": 0.0877
  }
]
```


The result has the EAN code, Product Category and Product Name. Additionally, Support and Confidence statistics are also returned for further analysis.

The following six files are required to build the flask API, 5 Python modules and 1 HTML module

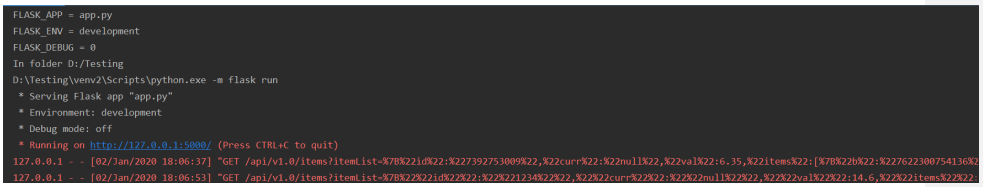
1. app.py: This python script contains the main entry point which when called uses other python scripts to map the requested data to map with the rule file.
2. error_handling: This script contains some classes to handle exception and will show custom error messages, but according to the new project requirements we are not calling this file and just giving default error messages.
3. files upload: This script is used to upload three required datasets.
    a. Processed_Product.txt: maps the EAN code of products to get its name
    b. Rules_DataFrame.txt: rule file
    c. Single_rule.txt: rule file
4. parsing -> this python script contains some important classes to process the JSON format data and Map it to other files.
5. recommend_DataFrame: This script processes the data, cleans it and maps it to the rules file.
6. Welcome.html

To run and test the API, following steps were taken:

1. Create a new Python project and import the following python libraries:

Package	Version	Latest version
Click	7.0	7.0
Flask	1.1.1	1.1.1
Jinja2	2.10.3	2.10.3
MarkupSafe	1.1.1	1.1.1
Werkzeug	0.16.0	0.16.0
itsdangerous	1.1.0	1.1.0
numpy	1.18.0	1.18.0
pandas	0.25.3	0.25.3
pip	19.0.3	▲ 19.3.1
python-dateutil	2.8.1	2.8.1
pytz	2019.3	2019.3
setuptools	40.8.0	▲ 44.0.0
six	1.13.0	1.13.0

2. Run the **app.py** script which creates the callable API



3. The blue text in the image http://127.0.0.1:5000/ is the port number on the local machine to access the API. The port number will be different for a different web server.

Machine Learning Framework

In general, any machine learning project poses some unique requirements that differ from more traditional data processing. The ML data lifecycle can be broadly said to comprise of four main parts.

1. Data Management
2. Model Learning
3. Model Verification
4. Model Deployment

Often these different stages may be performed by different teams of people, making it unfeasible for those who lack the resources to employ multiple teams. It is common to conduct ML experiments across different mixes and matches of data and features. In any highly experimental

process, it is essential that one can reproduce the results as needed. Users are traditionally forced to manually track the dependencies among data versions, training tasks, and ML models etc.

To streamline this the team implemented a working framework to manage the machine learning life cycle including data ingestion → data cleaning → model development → model evaluation → model deployment → model consumption. This is based around the open source software MLFlow.

Figure 21 shows a high-level schematic of the framework. MLFlow has multiple functionalities to help with the ML process, the most important for us is a tracking server.

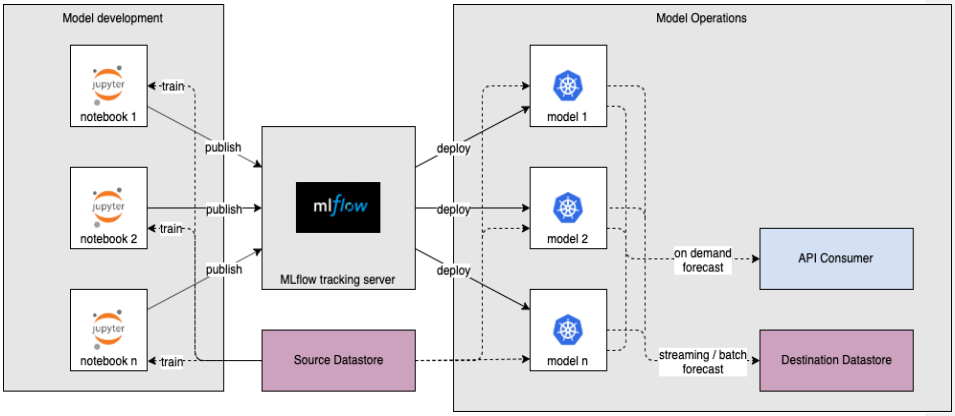


Figure 21: Schematic of the ML model framework

The ML model in question is written using custom Python code. Since every problem in different domains require different approaches or models, we have created a custom Python package. This contains some helper functions to simplify the whole process to ease machine learning development for new data scientists.

When the models are trained and tested, the tracking server keeps track of the parameters and model values to ease picking the best model. The model can then be deployed into the Glantus infrastructure using custom APIs.

The main model developed for deployment in the Glantus infrastructure was a Market Basket Analysis recommender system. Glantus provided point of sale data on 2,000,000

transactions from a UK based retail chain. This “basket” data contained thousands of items which needed to be identified and matched via JSON field in the various product files. This pre-processing JSON parsing was done in python to create a list of transactions for the MBA.

Figure 15 shows a brief overview of the entire process from the raw transaction data to the deployment of the API. The frequent pattern growth (FPGrowth) algorithm was chosen to perform the MBA. The algorithm mines the data and forms what are called frequent item sets, those that appear sufficiently often. From these, it creates rules which encompass the information on whether somebody is likely to buy certain items with others. After generating the rules, the resultant recommendations are implemented in the Glantus framework via custom API.

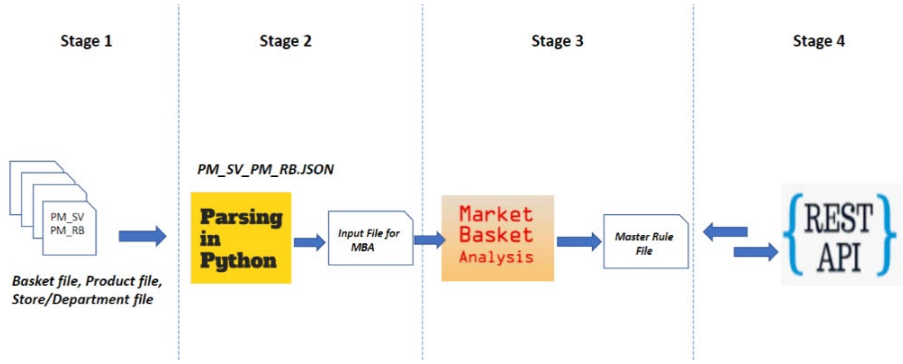


Figure 22: Market Basket Analysis Schematic

A number of other models have also been implemented into the Glantus framework, including a regression model to predict future liquidity ratings for banks using macroeconomic data and a model for predicting energy usage across a country.

Voucher Redemption Prediction

This section describes the additional analysis of voucher redemption using the data from the marketing department of a UK-based retailer. The retailer decides on the design, delivery and the target group for the voucher. This is then passed to a second organization which carries out the distribution and monitoring. The characteristic of the voucher is determined

from existing beliefs and historical results the marketing professionals have. Yet, some voucher campaigns can be more successful than others because no two are designed or executed in the same set of circumstances.

Success of a voucher campaign has, at least in the literature, been measured by the proportion of vouchers which are redeemed. However, the level of redemption rates is not the full story. Since one purpose of vouchers is about increasing sales, this can be achieved by different numbers of vouchers issued, so redemption can be an inaccurate measure. In order for a voucher campaign to be successful, the reduced margin on the product should be offset by the increased basket margin. Study suggests that 40% of shoppers would spend more overall and 28% would purchase a more expensive product with an exclusive offer. Similarly, 37% of voucher users spend more than shoppers who do not use them. Therefore, if we can calculate the expected increase in basket value of voucher shoppers, then we can determine the overall increased revenue. Manufacturers also issue vouchers through supermarkets, one of the reason is for increased brand loyalty in order to hold market share. However, it still impacts the supermarket in the same way. In this paper, it is the supermarket perspective which we will follow.

Data

The available data covers a five-day period in 2018 from a supermarket chain in the UK. The key part of the data is a set of transactional records. These transactions or baskets indicate the products bought, the overall value, location and the time, also importantly, whether there was a voucher used and if so, from which campaign it came. These transaction tables are the basis for calculating the success of a campaign (either redemption rate or increase in basket value). Additional supporting data used for the factors in the model comes from the store, product and campaign tables.

With the data available we can address some of the factors in the literature, apart from the personal data of the shopper. Our approach through machine learning is more general, as it can be used with general retail data.

Factors Chosen

The following seven factors are selected for our research based on, 1. available data, 2. literature review and 3. discussions with marketing people at the retail organization. The factors can be divided into those determined at the voucher design stage and those related to the transaction. The time of year would also be considered as a factor, but since our data is restricted within one week, it is not considered.

Transaction related factors

Region (north, central, north west, central & eastern, central)
Certain regions of the country may have greater proneness to redeeming vouchers because of their demographic composition. In other words, a region may represent more strongly certain demographic attributes such as age/sex/ethnicity.

Store Format (Petrol, Food, Market Town, Convenience)
Statistically, some venues do not attract vouchers – this could be because of the nature of the engagement between customer and shop, such as a petrol station which is short and predominantly involves the purchase of fuel. It could also be because of its price band – more expensive items give a greater payback, yet may attract a demographic less prone to using vouchers.

Store Size (Small, Medium, Large)
Plausibly, large shops have greater range of items and so may match better a wider range of vouchers.

Voucher related factors

Product Category (beverages, bakery, canned goods, dairy, dry baking, frozen, meat, produce, cleaning, paper, personal care, other)
The type of product could play some part in a voucher being redeemed or not. If it is a product in a category of commonly used items, such as bakery or toiletries, then it may have more traction than others.

Type (absolute, percentage, 'buy X then get Y')

Although all vouchers give some sort of benefit, they can do in different ways. This is one of the basic design decisions of the voucher.

Validity period (week, month)

The longer the validity period of the voucher, the greater the number of opportunities to use it. In our data, the validity period falls into two categories of around a week or a month.

Success Level (high, medium, low)

There are two levels of success we evaluate in this paper. The first is the tradition voucher redemption level and the second a more detailed financial outcome, related to improved revenues. For the redemption level, we compare the number of issued vouchers with the number redeemed, within the time window of the data. We chose two categories of relative success, based on the average redemption rates. The absolute values are quite small as the data was limited to only 5 days. The second level of success is a measure of the increased average basket value. In order to calculate this, we first establish the average basket value at a particular store from the non-voucher transaction database. Next, for each voucher and store we determine the average basket value of the voucher transactions. The weighted average increase/decrease gives a financial measure and hence a level for the success. Again two levels were adopted, corresponding to an increase or decrease to the average basket value.

Comparing the two measures, we see that the former seems less accurate given that vouchers can be issued ad hoc. Yet in practice, the number of vouchers are matched to relevant markets. The latter measure, can also be inaccurate, since we are comparing voucher users with non-voucher users in respect to average basket value. They can come from different populations. Yet, it gives us some measure of relative revenue success. With personal information we could actually measure how much more (or less) each individual voucher user spends when using a voucher.

A set of 540 transactions with vouchers were used in the analysis. A range of common prediction algorithms were applied, representing a cross section of approaches. These were Naïve Bayes, J48 decision tree, SMO support vector machine, IBK-1 & 2 nearest neighbour

and the OneR single rule. The software used was Weka [20] and accuracy was measured as the percentage of correctly predicted instances.

Results

The results for the voucher success problem, across six different prediction algorithms are shown in Tables 1 and 2.

Table 3. Accuracy across different prediction models with redemption outcome.

Evaluation criteria	Prediction algorithm								
	NB						SMO	IBK K=1	IBK K=2
Accuracy (%)	75		72	72	67	76			
	OneR	DT J48							
	74								

Table 2. Accuracy across different prediction models based with basket value outcome.

Evaluation criteria	Prediction algorithm								
	NB						SMO	IBK K=1	IBK K=2
Accuracy (%)	67		66	67	62	63			
	OneR	DT J48							
	64								

The results show that the different models, corresponding to different success measures, were significant in terms of prediction performance. Therefore, to predict voucher success, we can expect a reasonable level for the indirect redemption measure, rather than the more

direct revenue one - whatever the prediction algorithm that is chosen. There is little difference between classifiers for each set of data.

Conclusions and Future Work

In the history of voucher-led promotions, the redemption rate has been seen as the indicator of the campaign being successful or not. With more advanced retail information from tills and loyalty schemes, we can begin to advance the success criteria towards a more direct measure of increased revenue. However, the problem remains the same, predicting what campaign will be successful in light of past experience.
We have shown in this research, that prediction based on the traditional measure (non-personal) is quite accurate, whereas a measure, that seems closer to a revenue value, is not. Both are surrogate measures, just that one seems closer to the bottom line than the other. Certainly, the factors used have been taken from the former domain, but equally, if the campaign is well understood, then the simple redemption rate could suffice in measuring profitability.

This means that retailers now can have a more accurate, but perhaps indirect, measure of success than one which tries to measure campaign profitability. Certainly knowing more about the customer may help raise the accuracy level of the second model (as well as the first), but a more accurate measure for increased revenue may be better still.

Summary of the research paper (Fruct28 conference)

The products were clustered based on their frequency and price. Another important aspect of this study was to find interesting rules by performing differential market basket analysis to identify association rules. When using a cluster-based approach, it was observed that the same set of rules can be generated by using only 7% of the total 16210 items, which in turn directly contributes to reducing the processing overheads and thus reducing the computation time. Furthermore, results obtained from differential market basket analysis have highlighted a few interesting rules which were missing from the original set of rules. A clustering-based approach used in this study not only consists of frequent items but also

considers their contribution to the overall revenue generation by considering its price. In addition to this, the least contributing product exclusion rate is also improved from 45% to 93%. These results evidently suggest that the computation cost can be significantly reduced, and more accurate rules can be generated by applying differential market basket analysis.

Summary of IPP Final Results

- A new data science competency has been set up in the Glantus ecosystem comprised of an API from Azure cloud, in house Docker container and ML operations.

- Industry level working framework designed to manage machine learning life cycle (data ingestion → data cleaning → model development → model evaluation → model deployment → model consumption).

- A recommender system (RecSys) has been developed and integrated both in cloud (Azure) and in house (MLFlow, Docker/container) using Glantus' provided product and market basket data.

- A generic python machine learning package developed (based on Glantus infrastructure and compliance) to ease machine learning development for new data scientist.

- Onboarding process and steps have been setup for third party developed model consumption by Glantus' platform using newly built framework.

- ML models, RecSys, classification, time series, have been developed and evaluated by NCI data science team and integrated with framework.

Future work and improvement

- The generic data pre-processing should be set up and ‘single touch point’ where training data input file should be created for any model.
- The proper user role can be defined so to avoid confusion among data scientist, model monitoring, model deployment, model user etc.
- Testing the whole lifecycle of the model using more (size) data to check the performance in terms of data ingestion, data mapping, model training, deployment, output result.
- Make use of promopay data - Future buying/promotions to run - Predictions using basket data, vouchers used, shop locations\type - enhanced with location economic\social\demographic and weather data
- Advance the association rules for fraud detection scenarios, insurance industry
- Better integration with model development/notebooks (Ability to have integrated notebook for data prep & modelling i.e. the data science and ML workspace)
- Create 'Base' model notebook for different algorithms, allow user to select and new instance of notebook created with all references
- Notebook management and notebooks appearing within platform (easy model collaboration & re-use)
- Integration with MLFlow projects for better parameterization of model development
- Model promotion & rollback strategies/orchestration from platform
- For data scientists & large data management - Platform integration with 'Data Lake' & Data warehouse.

National College of Ireland

Research Team

Dr Anu Sahni
Principal Investigator

Dr Pramod Pathak
Co- Principal Investigator

Dr Paul Stynes
Project Manager

Dr Cian O'Hara
Post Doctorate Researcher

Pramit Kumar
Technical Lead

Arjit Agrawal
Senior Developer

Dr James Little
Senior Data Scientist

Nilay Kumar
Consultant

Abhishek Jain
Developer

Manus McDonald
Consultant

Technology Transfer Team

Bertie Kelly
Commercial Manager
NCI

Dr Andrew Marsh
Senior Licensing Executive
Hothouse, TU Dublin

Glantus Team

Geoff Keating
Chief Technology Officer

Marie O'Toole
Team Lead