itida
Developing IT in Egypt

Ministry of Communication and Information Technology

NTI
National Telecommunication Institute

Next Technology Leaders

Intensive programs on Communication and Information Technology (3 Months)

Track
Big Data Science

Recommendation Application

BY

Mahmoud Abdullah Tawfik

Acknowledgements

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. We would like to extend our sincere thanks to all of them.

We are highly indebted to our supervisors: Prof. Doaa Hassan and Eng. Sayed for their guidance and constant supervision as well as for providing necessary information regarding the project and also for their support in completing the project.

We would like to express our gratitude towards members of the National telecommunication Institute for their kind co-operation and providing us with necessary equipment for learning Data science Track.

Our thanks and appreciations also go to our colleagues in developing the project and people who have willingly helped us out with their abilities.

M.Aboalarbe, Mahmoud, Abdelrahman, Mohammed

Abstract

In this project, we explore the technique of recommendation engine and its different types on Market basket to recommend items of the online Market to the most desirable customers. Using a dataset obtained from Instacart.com, we apply Association rule with Apriori Algorithm of about 7000000 rows. We compare performance time of the project when it run with data science and with Apache spark using databricks.com, then we make an API model and connect it with a pretty graphical user interface in an Android Application to be more friendly to the users.

List of figures

Table of Contents

Chapter one

Introduction

Chapter One

Introduction

1.1 project main idea

The main idea of the project is to make a recommendation application to recommend grocery product for customers so the market could save its money spent advertising and customers save their time for searching for their desired products.

1.2 problem Statement

Companies spent money on advertising and sometimes high portion of this money doesn't make an effect because these advertising go to the wrong customers that don't prefer these products or don't match their needs, so we want to make an application that would recommend products to the most desirable Customers that have high probability to buy these items.

1.3 proposed solution

Our solution is to make a recommendation application that recommend the grocery items to the customers according to their needs, so company can recommend the best products for their customer that have high probability to buy these products.

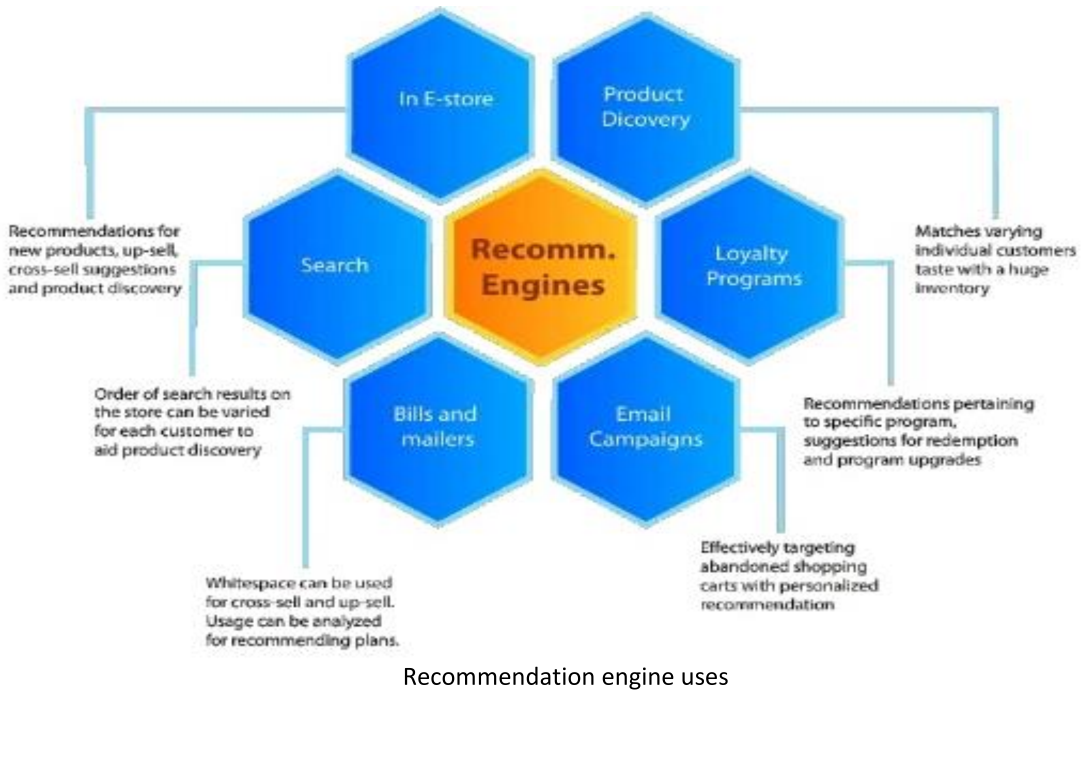
Chapter Two
Background and Theoretical Overview

Chapter Two

Background and Theoretical Overview

2.1 Scientific background

2.1.1 Recommendation Engines


In E-store
Product Dicovery
Search
Recomm. Engines
Loyalty Programs
Bills and mailers
Email Campaigns
Recommendations for new products, up-sell, cross-sell suggestions and product discovery
Matches varying individual customers taste with a huge inventory
Order of search results on the store can be varied for each customer to aid product discovery
Recommendations pertaining to specific program, suggestions for redemption and program upgrades
Effectively targeting abandoned shopping carts with personalized recommendation
Whitespace can be used for cross-sell and up-sell. Usage can be analyzed for recommending plans.

Recommendation engine uses

In today’s world, every customer is faced with multiple choices. For example, If I’m looking for a book to read without any specific idea of what I want, there’s a wide range of possibilities how my search might pan out. I might waste a lot of time browsing around on the internet and trawling through various sites hoping to strike gold. I might look for recommendations from other people.

But if there was a site or app which could recommend me books based on what I have read previously, that would be a massive help. Instead of wasting time on various sites, I could just log in and voila! 10 recommended books tailored to my taste.

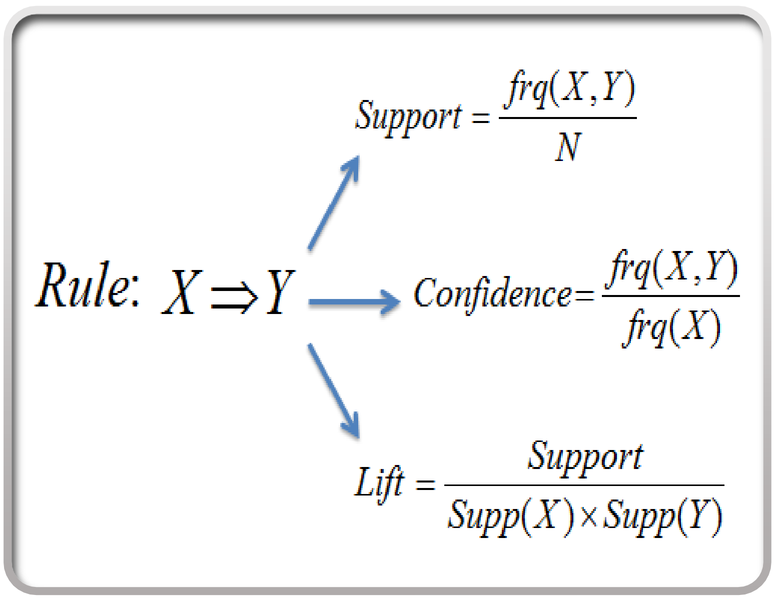
This is what recommendation engines do and their power is being harnessed by most businesses these days. From Amazon to Netflix, Google to Goodreads, recommendation engines are one of the most widely used applications of machine learning techniques. A recommendation engine filters the data using different algorithms and recommends the most relevant items to users. It first captures the past behavior of a customer and based on that, recommends products which the users might be likely to buy.

If a completely new user visits an e-commerce site, that site will not have any past history of that user. So how does the site go about recommending products to the user in such a scenario? One possible solution could be to recommend the best-selling products, i.e. the products which are high in demand. Another possible solution could be to recommend the products which would bring the maximum profit to the business.

If we can recommend a few items to a customer based on their needs and interests, it will create a positive impact on the user experience and lead to frequent visits. Hence, businesses nowadays are building smart and intelligent recommendation engines by studying the past behavior of their users.

2.1.2 Association rule

Association Rules find all sets of items (itemsets) that have support greater than the minimum support and then using the large itemsets to generate the desired rules that have confidence greater than the minimum confidence. The lift of a rule is the ratio of the observed support to that expected if X and Y were independent. A typical and widely used example of association rules application is market basket analysis


$Rule: X \Rightarrow Y$
$Support = \frac{frq(X,Y)}{N}$
$Confidence = \frac{frq(X,Y)}{frq(X)}$
$Lift = \frac{Support}{Supp(X) \times Supp(Y)}$

Association Rule

Support (S):

Support is an indication of how frequently the items appear in the database.

Confidence (C):

indicates the number of times the if/then statements have been found to be true.

Lift:

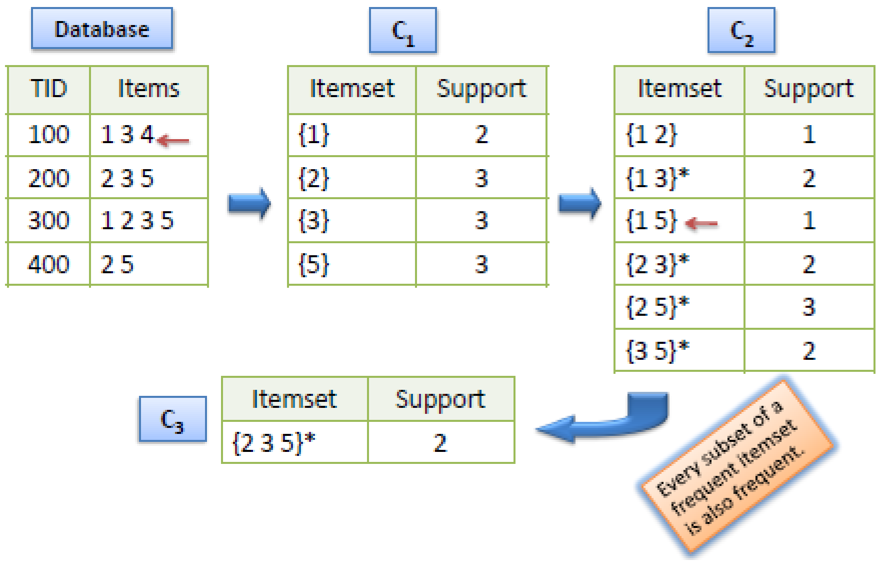
Lift is the ratio of the observed support to that expected if X and Y were independent.

2.1.3 Apriori Algorithm

1. Candidate itemsets are generated using only the large itemsets of the previous pass without considering the transactions in the database.

2. The large itemset of the previous pass is joined with itself to generate all itemsets whose size is higher by 1.

3. Each generated itemset that has a subset which is not large is deleted. The remaining itemsets are



Database

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

C_1

Itemset	Support
{1}	2
{2}	3
{3}	3
{5}	3

C_2

Itemset	Support
{1 2}	1
{1 3}*	2
{1 5}	1
{2 3}*	2
{2 5}*	3
{3 5}*	2

C_3

Itemset	Support
{2 3 5}*	2

Apriori algorithm

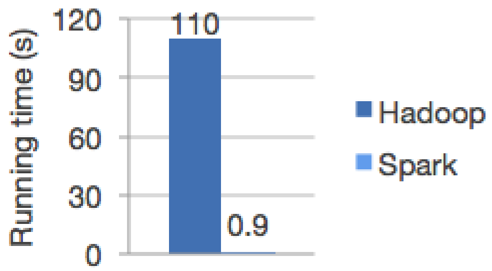
The Apriori algorithm takes advantage of the fact that any subset of a frequent itemset is also a frequent itemset. The algorithm can therefore, reduce the number of candidates being considered by only exploring the itemsets whose support count is greater than the minimum support count. All infrequent itemsets can be pruned if it has an infrequent subset.

2.1.4 Apache Spark

Apache Spark is an open-source distributed general-purpose cluster-computing framework. Originally developed at the University of California, Berkeley's AMPLab, the Spark codebase was later donated to the Apache Software Foundation, which has maintained it since. Spark provides an interface for programming entire clusters with implicit data parallelism and fault tolerance.

Advantages of Apache Spark

Speed
Run workloads 100x faster.
Apache Spark achieves high performance for both batch and streaming data, using a state-of-the-art DAG scheduler, a query optimizer, and a physical execution engine.



Logistic regression in Hadoop and Spark (Speed)

Ease of Use
Write applications quickly in Java, Scala, Python, R, and SQL.
Spark offers over 80 high-level operators that make it easy to build parallel apps. And you can use it *interactively* from the Scala, Python, R, and SQL shells.

Generality
Combine SQL, streaming, and complex analytics.
Spark powers a stack of libraries including SQL and DataFrames, Mllib for machine learning, GraphX, and Spark Streaming. You can combine these libraries seamlessly in the same application.

Runs Everywhere
Spark runs on Hadoop, Apache Mesos, Kubernetes, standalone, or in the cloud. It can access diverse data sources.

You can run Spark using its standalone cluster mode, on EC2, on Hadoop YARN, on Mesos, or on Kubernetes. Access data in HDFS, Alluxio, Apache Cassandra, Apache Hbase, Apache Hive, and hundreds of other data sources.

Apache Spark Uses

2.1.5 Android (operating system)

Android is a mobile operating system developed by Google, based on a modified version of the Linux kernel and other open source software and designed primarily for touchscreen mobile devices such as smartphones and tablets. In addition, Google has further developed Android TV for televisions, Android Auto for cars, and Wear OS for wrist watches, each with a specialized user interface. Variants of Android are also used on game consoles, digital cameras, PCs and other electronics.

Initially developed by Android Inc., which Google bought in 2005, Android was unveiled in 2007, with the first commercial Android device launched in September 2008. The operating system has since gone through multiple major releases, with the current version being 9 “Pie”, released in August 2018. The core Android source code is known as Android Open Source Project (AOSP), and is primarily licensed under the Apache License.

Android is also associated with a suite of proprietary software developed by Google, called Google Mobile Services (GMS) that very frequently comes pre-installed in devices, which usually includes the Google Chrome web browser and Google Search and always includes core apps for services such as Gmail, as well as the application store and digital distribution platform Google Play, and associated development platform. These apps are licensed by manufacturers of Android devices certified under standards imposed by Google, but AOSP has been used as the basis of competing Android ecosystems, such as Amazon.com’s Fire OS, which use their own equivalents to GMS.

Chapter Three
Project Implementation

Chapter Three
Project Implementation

3.1 Used Tools

Our tools that was used in this project:

- Python v3 programming language
- Android studio
- Python Flask for API on PythonAnyWhere
- Juypter Notebook
- Apache spark on Databricks

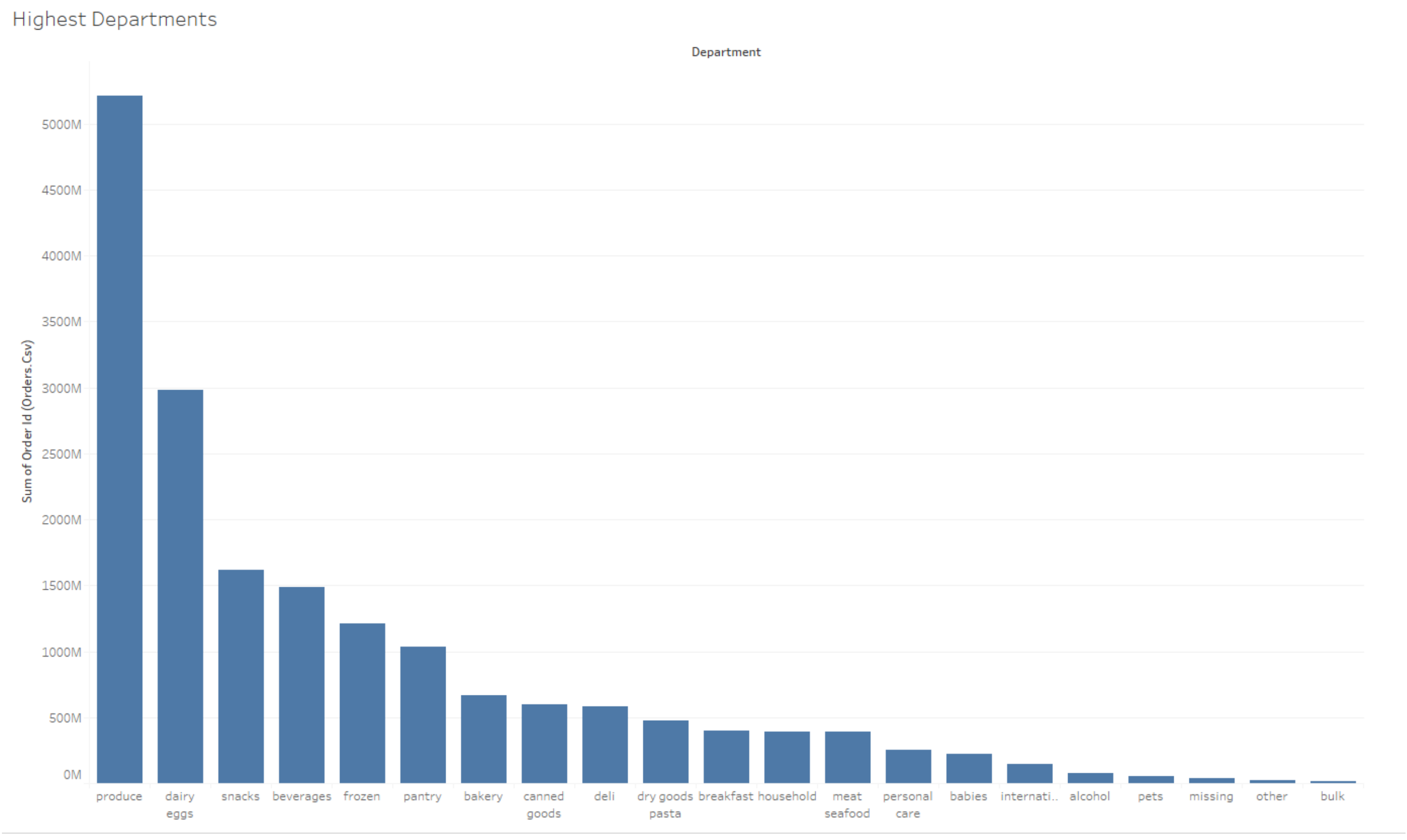
3.2 Implementation

3.2.1 Data set description

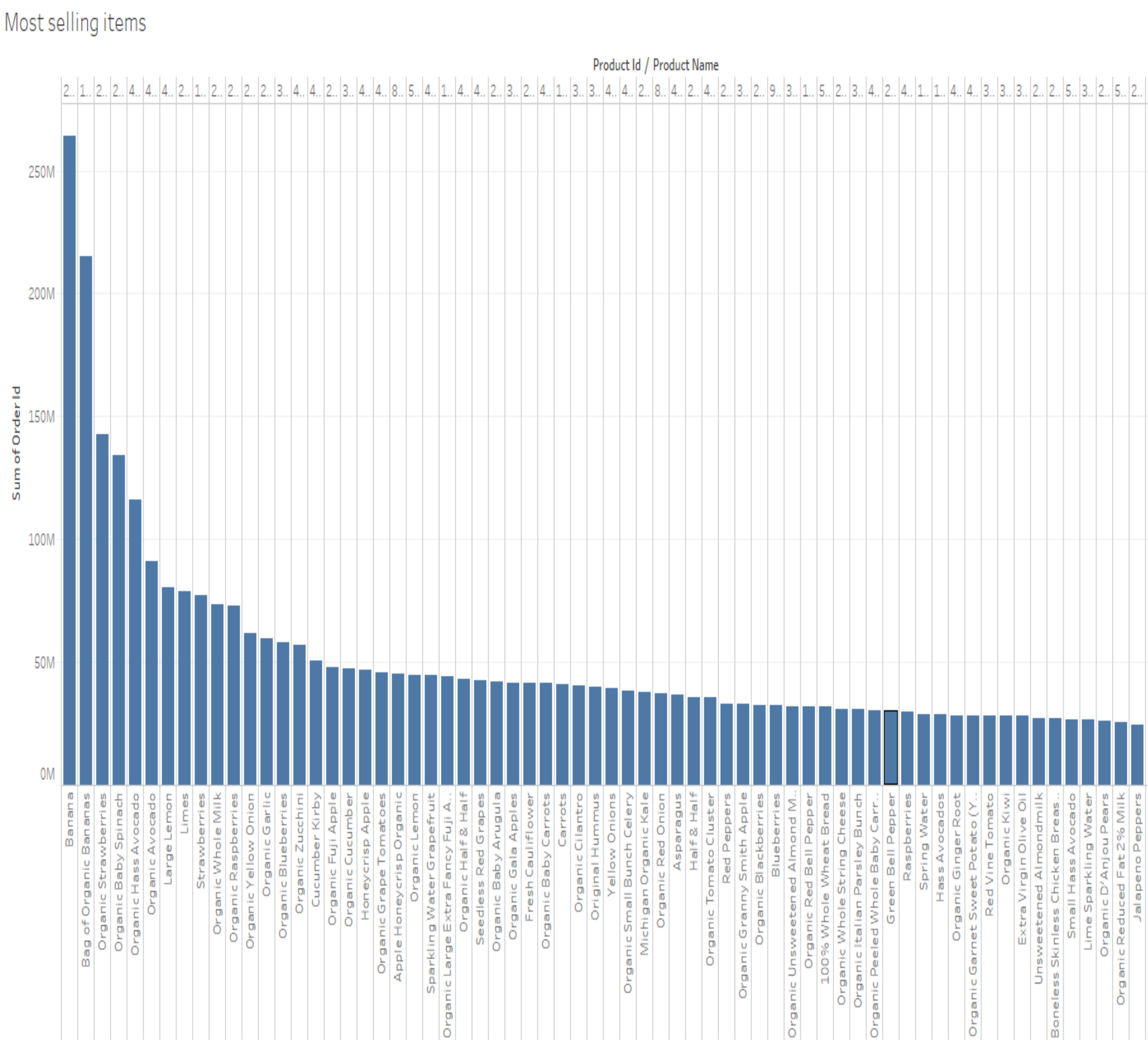
Our dataset consists of 32434489 rows and four features
The URL of dataset: https://www.instacart.com/datasets/grocery-shopping-2017

3.2.2 Data set visualization:

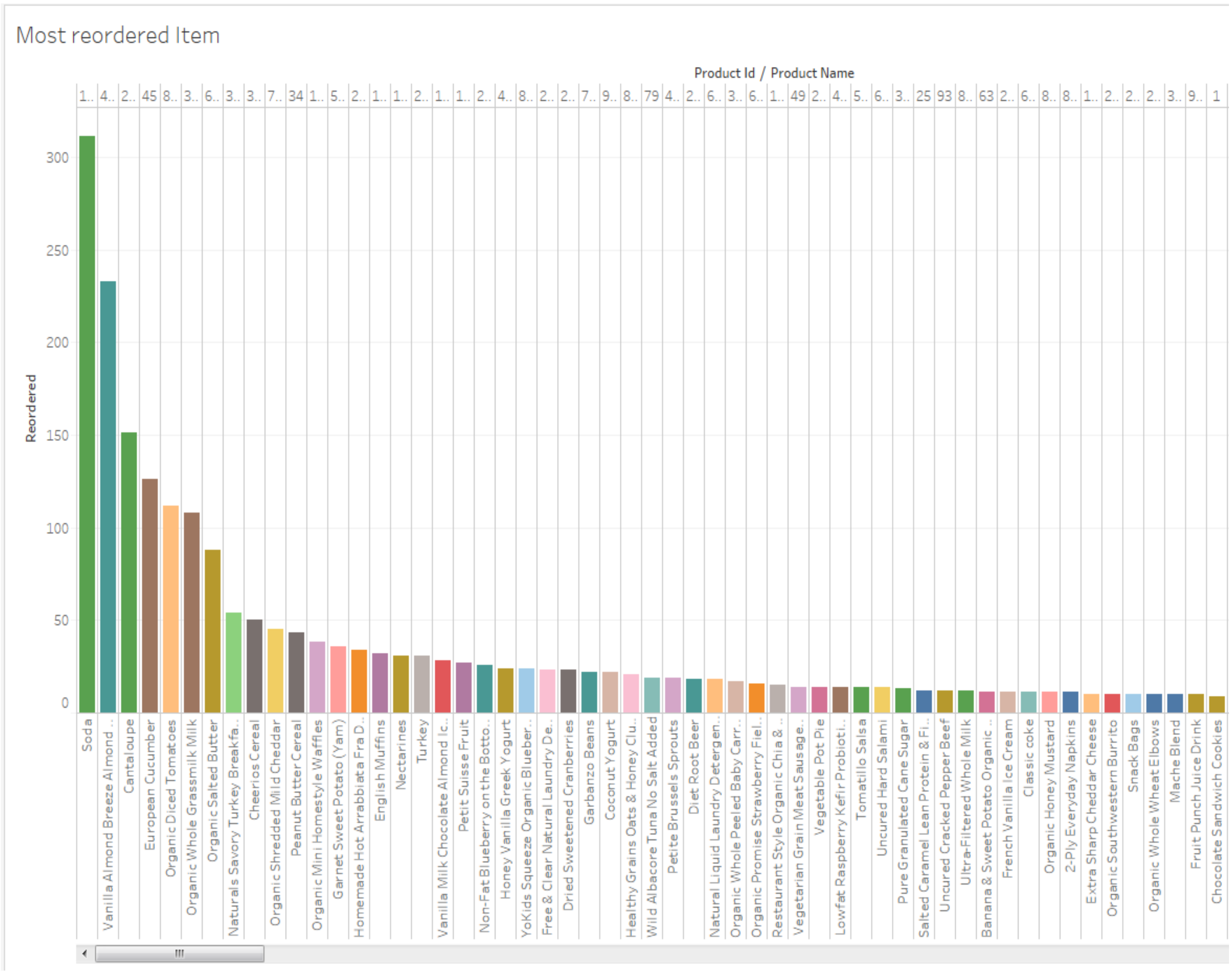
Using **Tableau,** we have reached some visualizations:



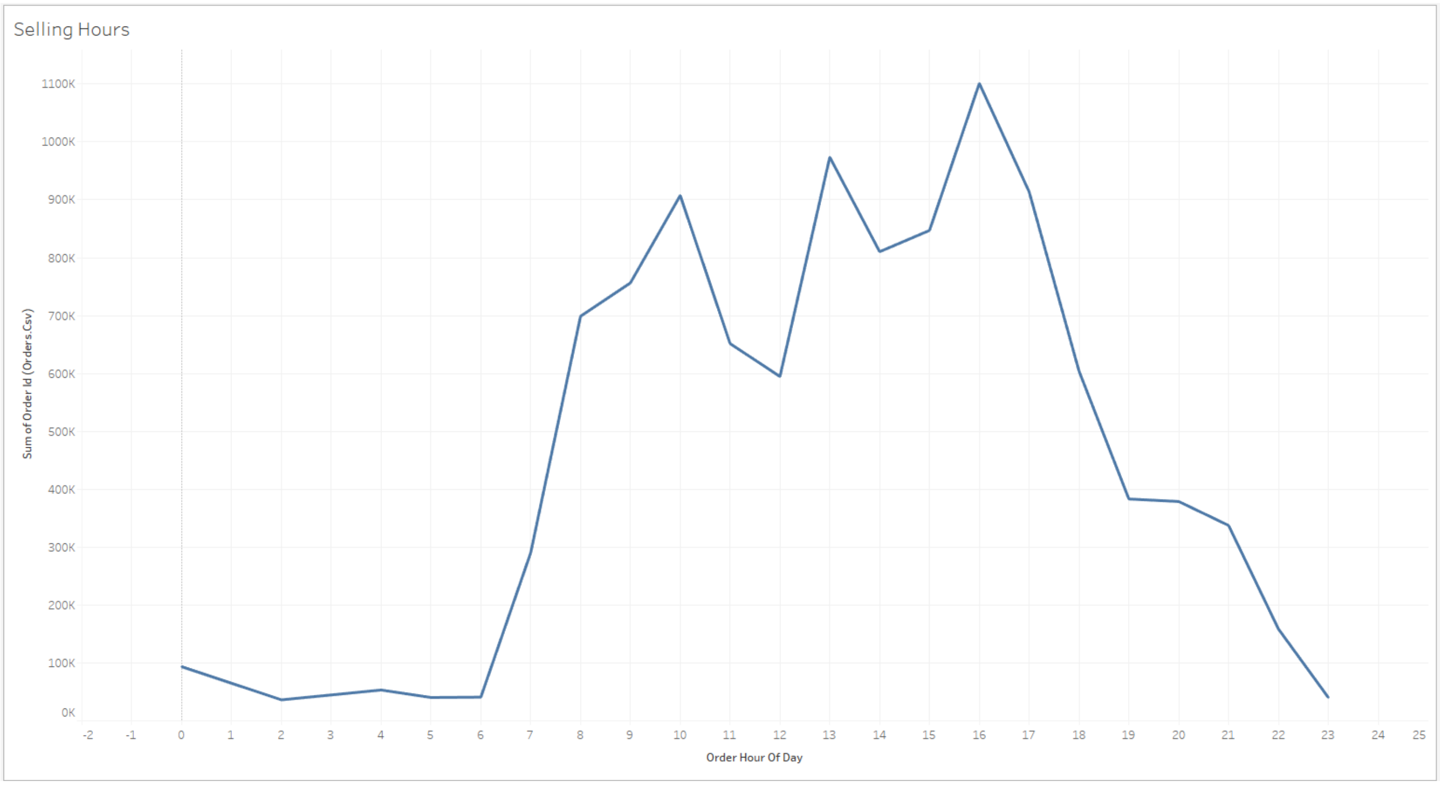
Highest department



Most sold items



Most reordered Item



Highest selling hour

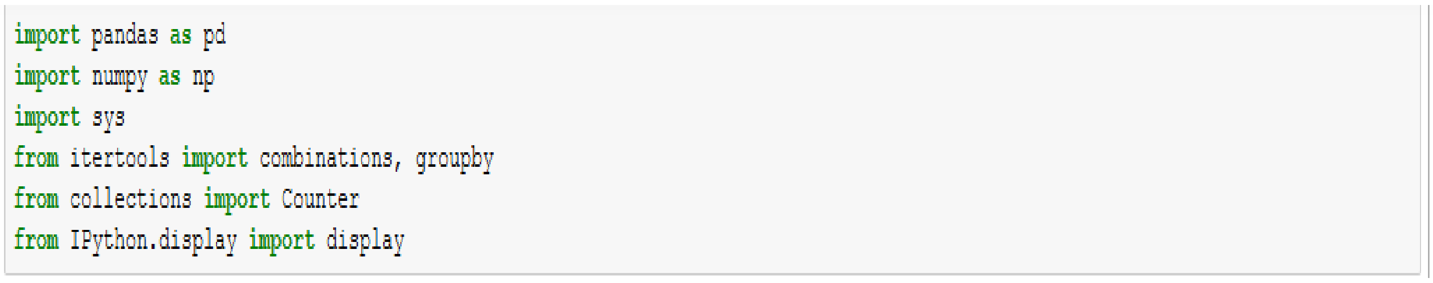
3.2.3 IN Data Science Model

- We have worked on dataset which consist of 7000000 rows and two features to be able to analyze and process this dataset using Juypter Notebook with python

We have used association rule for making this recommendation system with Apriori Algorithm

First**:**

we import libraries in our project

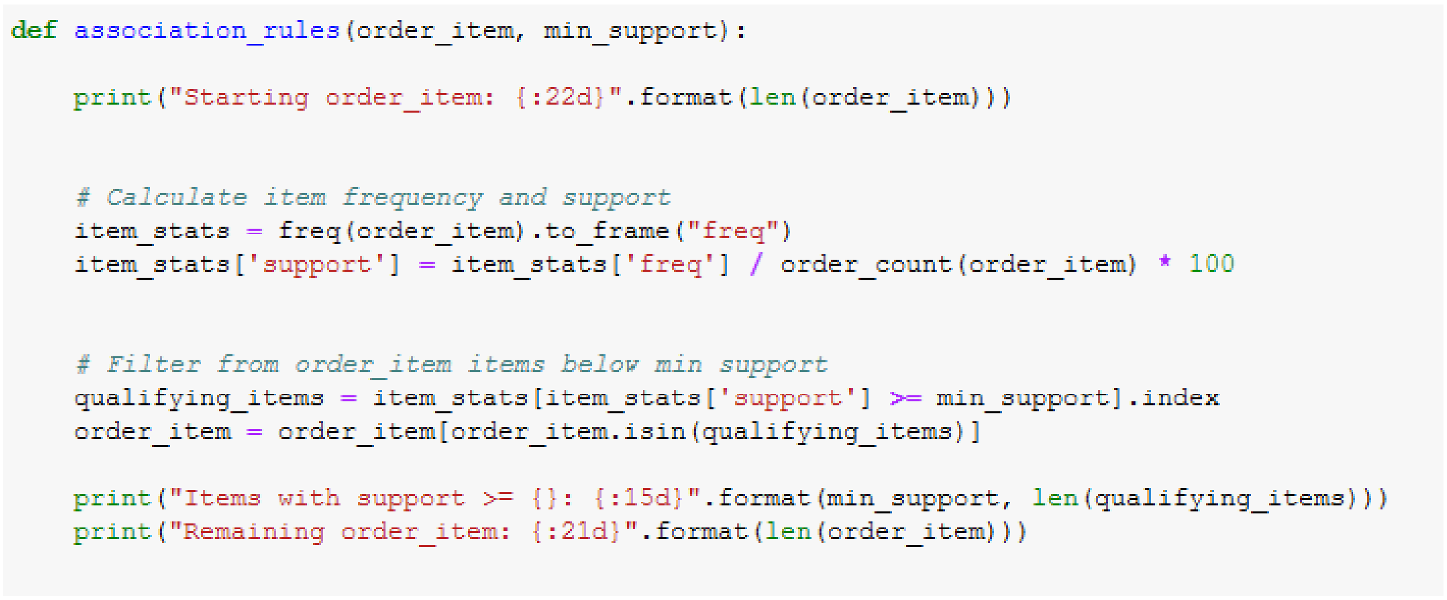


```
import pandas as pd
import numpy as np
import sys
from itertools import combinations, groupby
from collections import Counter
from IPython.display import display
```

import libraries

Second:

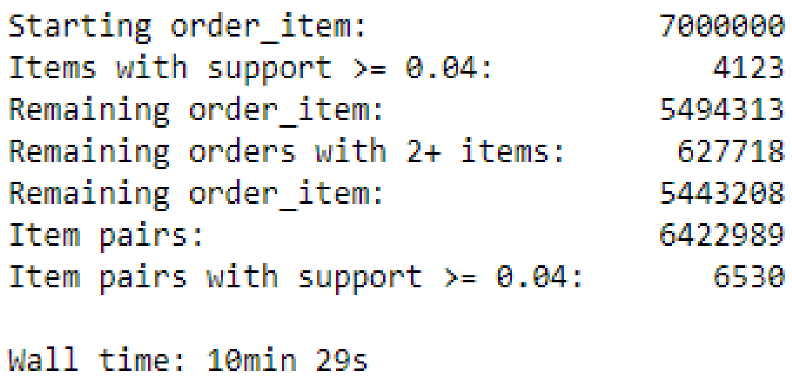
The main function of the system which uses association rule



```
def association_rules(order_item, min_support):

    print("Starting order_item: {:22d}".format(len(order_item)))


    # Calculate item frequency and support
    item_stats = freq(order_item).to_frame("freq")
    item_stats['support'] = item_stats['freq'] / order_count(order_item) * 100


    # Filter from order_item items below min support
    qualifying_items = item_stats[item_stats['support'] >= min_support].index
    order_item = order_item[order_item.isin(qualifying_items)]

    print("Items with support >= {}: {:15d}".format(min_support, len(qualifying_items)))
    print("Remaining order_item: {:21d}".format(len(order_item)))
```

association rule

After using Association rule, we reach this **result**:



```
Starting order_item:                 7000000
Items with support >= 0.04:             4123
Remaining order_item:                5494313
Remaining orders with 2+ items:       627718
Remaining order_item:                5443208
Item pairs:                          6422989
Item pairs with support >= 0.04:        6530

Wall time: 10min 29s
```

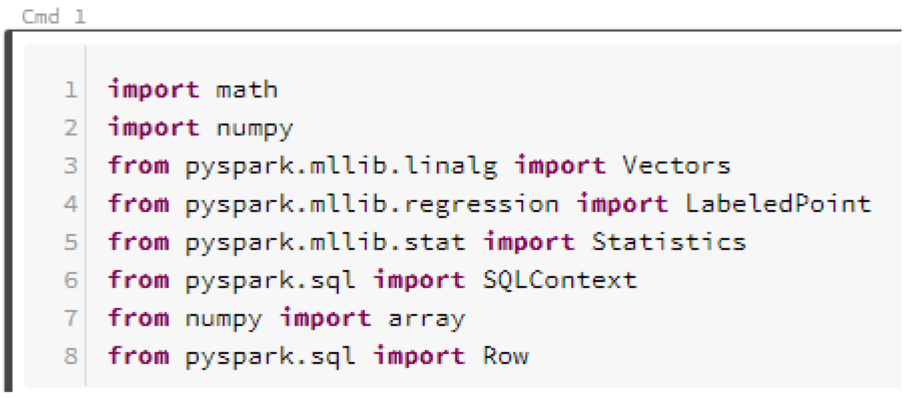
result

3.2.4 IN Big Data Model

We have worked on dataset which consist of 7000000 rows and two features to be able to analyze and process this dataset using Apache Spark on Databricks

First:

we import libraries in our project

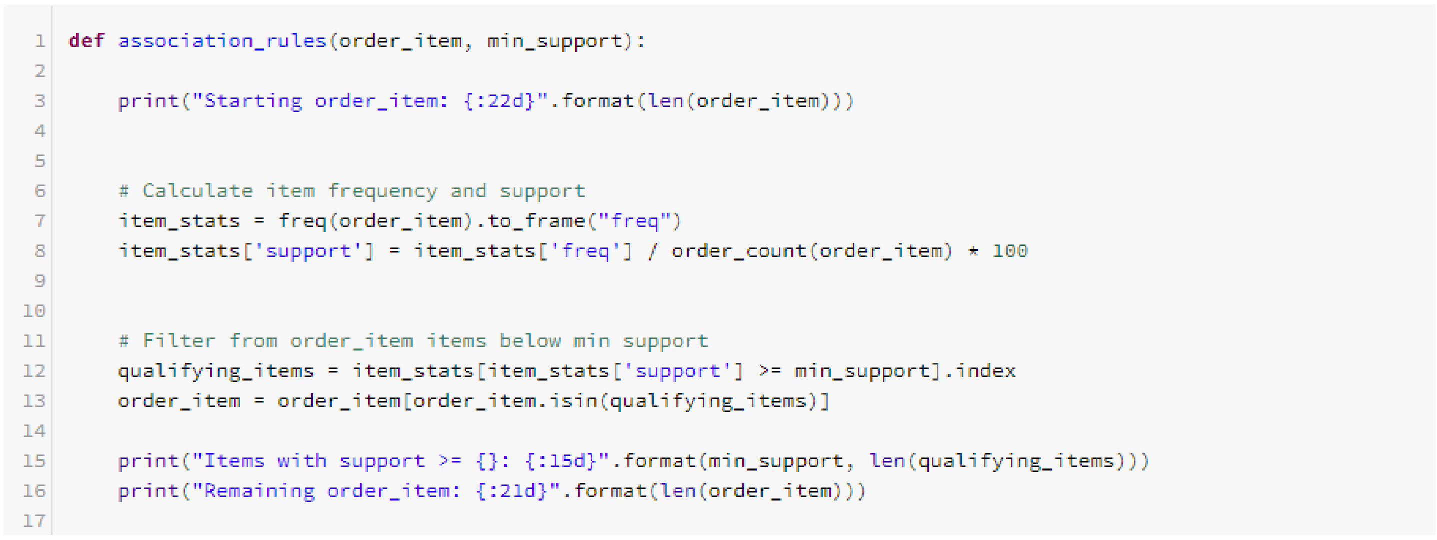


Cmd 1

```python
import math
import numpy
from pyspark.mllib.linalg import Vectors
from pyspark.mllib.regression import LabeledPoint
from pyspark.mllib.stat import Statistics
from pyspark.sql import SQLContext
from numpy import array
from pyspark.sql import Row
```

Second:

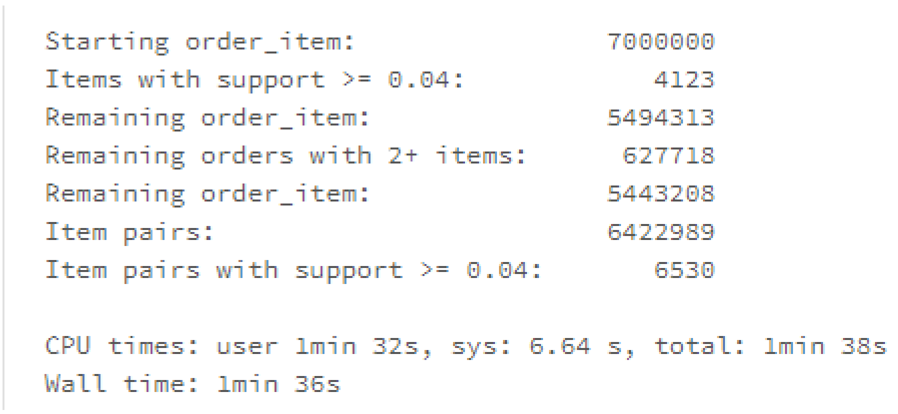
The main function of the system which uses association rule



```python
def association_rules(order_item, min_support):

    print("Starting order_item: {:22d}".format(len(order_item)))


    # Calculate item frequency and support
    item_stats = freq(order_item).to_frame("freq")
    item_stats['support'] = item_stats['freq'] / order_count(order_item) * 100


    # Filter from order_item items below min support
    qualifying_items = item_stats[item_stats['support'] >= min_support].index
    order_item = order_item[order_item.isin(qualifying_items)]

    print("Items with support >= {}: {:15d}".format(min_support, len(qualifying_items)))
    print("Remaining order_item: {:21d}".format(len(order_item)))

```

association rule

After using Association rule with Apache Spark on Data bricks we reach this **result**:

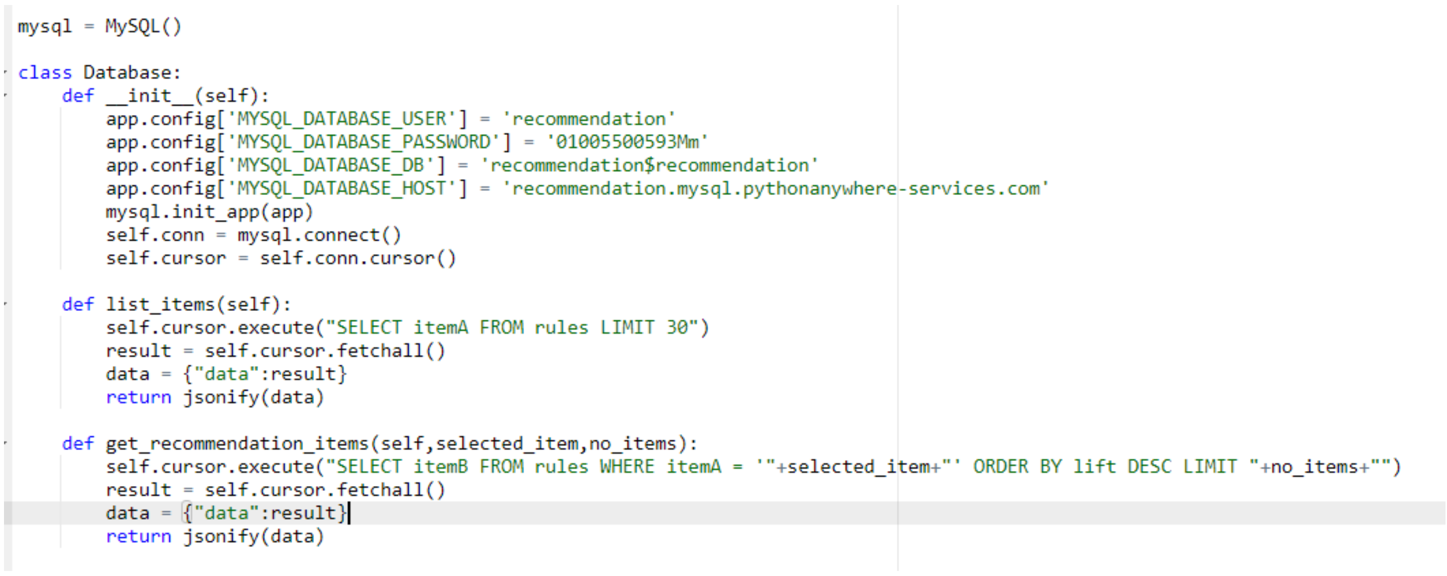


```
Starting order_item:                   7000000
Items with support >= 0.04:               4123
Remaining order_item:                  5494313
Remaining orders with 2+ items:         627718
Remaining order_item:                  5443208
Item pairs:                            6422989
Item pairs with support >= 0.04:          6530

CPU times: user 1min 32s, sys: 6.64 s, total: 1min 38s
Wall time: 1min 36s
```

result

3.2.5 API:

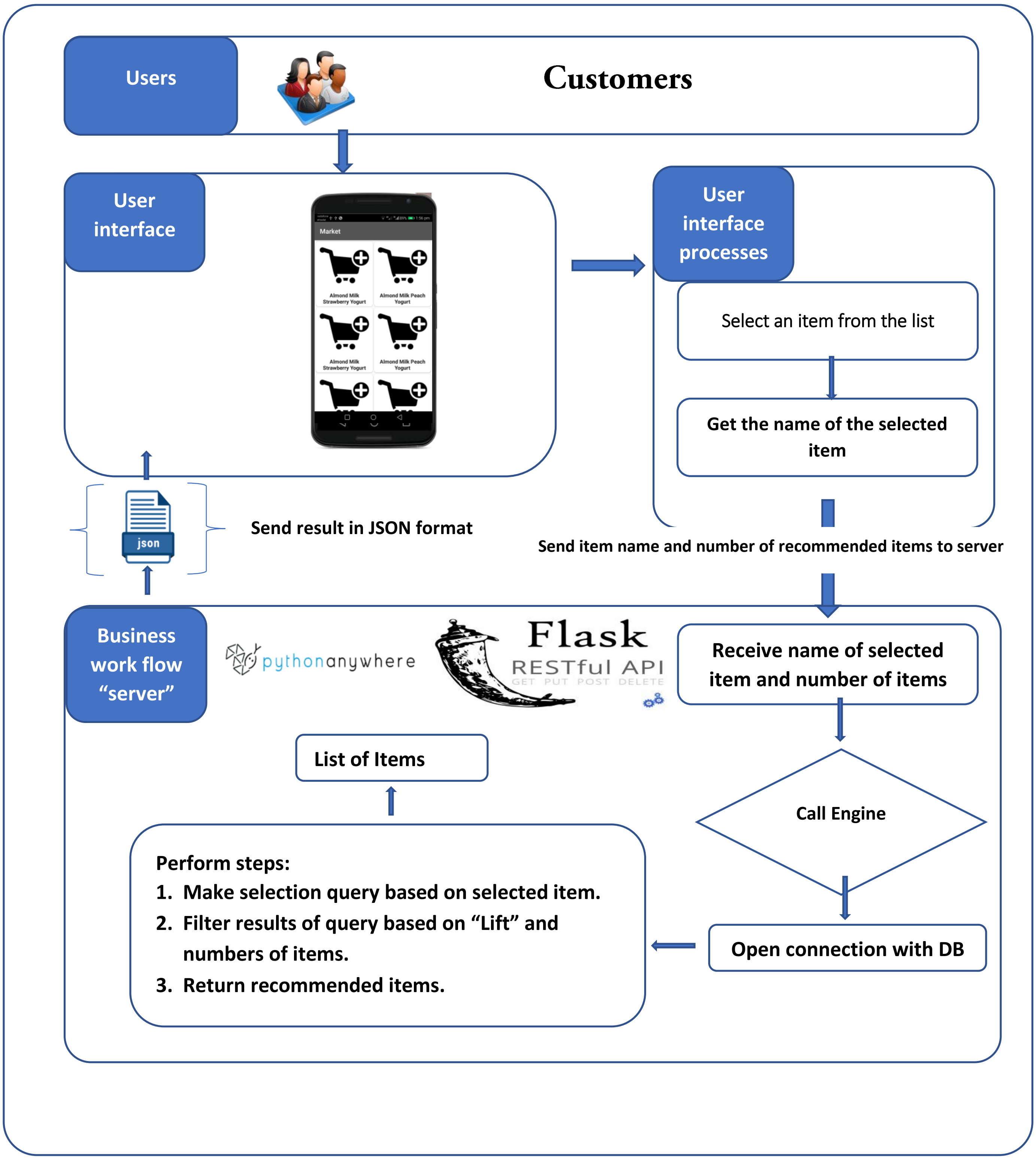
We use **PythonAnyWhere** is a rule-based machine learning method for discovering interesting relations between variables in large databases. It is intended to identify strong rules discovered in databases using some measures of interestingness. This rule-based approach also generates new rules as it analyzes more data. The ultimate goal, assuming a large enough dataset, is to help a machine mimic the human brain’s feature extraction and abstract association capabilities from new uncategorized data.



```
mysql = MySQL()

class Database:
    def __init__(self):
        app.config['MYSQL_DATABASE_USER'] = 'recommendation'
        app.config['MYSQL_DATABASE_PASSWORD'] = '01005500593Mm'
        app.config['MYSQL_DATABASE_DB'] = 'recommendation$recommendation'
        app.config['MYSQL_DATABASE_HOST'] = 'recommendation.mysql.pythonanywhere-services.com'
        mysql.init_app(app)
        self.conn = mysql.connect()
        self.cursor = self.conn.cursor()

    def list_items(self):
        self.cursor.execute("SELECT itemA FROM rules LIMIT 30")
        result = self.cursor.fetchall()
        data = {"data":result}
        return jsonify(data)

    def get_recommendation_items(self,selected_item,no_items):
        self.cursor.execute("SELECT itemB FROM rules WHERE itemA = '"+selected_item+"' ORDER BY lift DESC LIMIT "+no_items+"")
        result = self.cursor.fetchall()
        data = {"data":result}
        return jsonify(data)
```

API

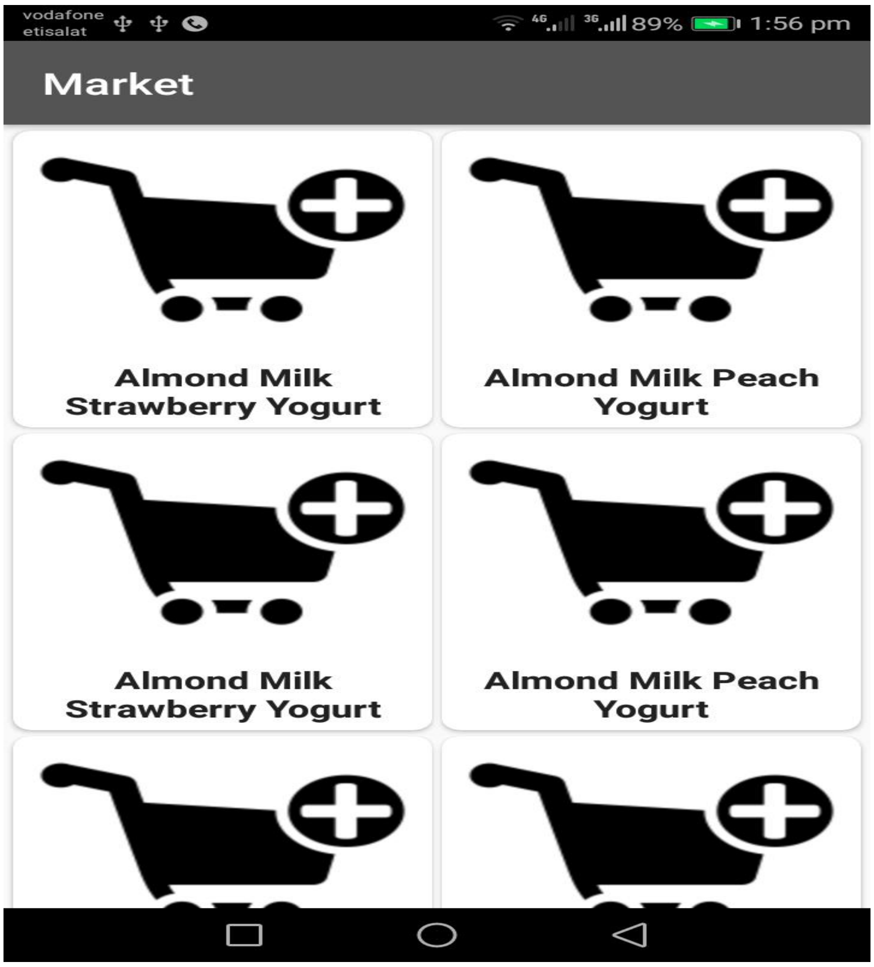
3.3 project scenario


Users
Customers
User interface
Almond Milk Strawberry Yogurt
Almond Milk Peach Yogurt
User interface processes
Select an item from the list
Get the name of the selected item
json
Send result in JSON format
Send item name and number of recommended items to server
Business work flow "server"
pythonanywhere
Flask
RESTful API
GET PUT POST DELETE
Receive name of selected item and number of items
List of Items
Call Engine
Perform steps:
1. Make selection query based on selected item.
2. Filter results of query based on "Lift" and numbers of items.
3. Return recommended items.
Open connection with DB

project scenario

3.4 project Demo

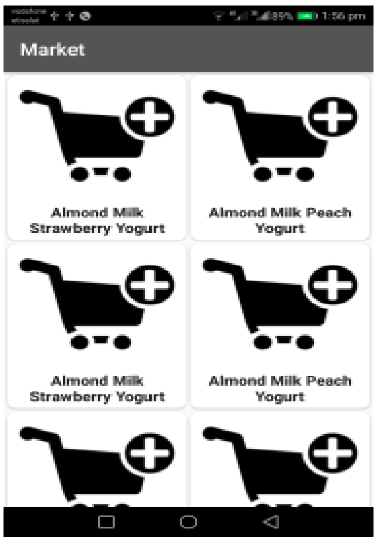
We Use Graphical User Interface (GUI) using Android Application with friendly interface to the customers showing all the items and make him to choose item and then Application would recommend to the user the item based on his choice.



GUI in Android APP

First:

We make Android Application with friendly Graphical user interface that is item-based recommendation engine in background and recommend items to the user.



GUI in Android APP

Second:

We measure run-time and see difference with Apache Spark on big data, python on data science and cloudera and this was the result:

	Processing Time	No. of data items
Python	10 min 29 sec	7000000
Apache Spark	1 min 36 Sec	7000000
Apache Spark	2 min 11 Sec	10000000
Apache Spark	CPU Hanging	10000000
Cloudera VM	1 min 35 Sec	7000000

We have run the algorithm on local machine of the following properties:

- Operating System: Windows 10 Pro 64-bit (10.0).
- Processor: Intel(R) Core(TM) i5-2520M CPU @ 2.50GHz (4 CPUs), 2.5GHz.
- Memory: 6G RAM.
- Display Memory: 1664 MB.
- Shared Memory: 1632 MB.

References:

- https://www.saedsayad.com/association_rules.htm
- https://en.wikipedia.org/wiki/Association_rule_learning
- Data Mining concepts and techniques by Jiawei Han
- https://en.wikipedia.org/wiki/PythonAnywhere
- https://en.wikipedia.org/wiki/Machine_learning
- https://www.adobe.com/experience-cloud/topics/recommendation-engine.html
- https://towardsdatascience.com/a-gentle-introduction-on-market-basket-analysis-association-rules-fa4b986a40ce