INTELLIGENT SHOPPER

TE CSE Mini project

By

GR NO.	NAME	ROLL NO.
141438	**Christine Abraham**	**TY-E-2**
141541	**Amitabh Saini**	**TY-E-4**
141542	**Nidhi Saini**	**TY-E-51**
141357	**Omkar Kashid**	**TY-C-61**

Under the guidance of

Prof. Dr. Ketan Bahulkar

Department of Computer Engineering

BRaCT'S Vishwakarma Institute Technology

PUNE – 411 037

Bansilal Ramnath Agarwal Charitable Trust's

Vishwakarma Institute Technology

PUNE – 411 037

CERTIFICATE

Certified that this Project report titled "**Intelligent Shopper**" by "**Omkar Kashid (TY-C-61), Christine Abraham (TY-E-2), Amitabh Saini (TY-E-4), Nidhi Saini (TY-E-51),**" is approved by me for submission. Certified further that, to the best of my knowledge, the report represents work carried out by the students as the Mini project as prescribed by the **Savitribai Phule Pune University** in the academic year **2016-17**.

Guide	**Head of Department**
(Prof. Dr. Ketan Bahulkar)	**(Prof. Dr. Vivek Deshpande)**

Date:

ACKNOWLEDGEMENT

We take this opportunity to express our profound gratitude and deep regards to the Director, Prof. (Dr.) R.M.Jalnekar for giving us this golden opportunity of doing a project as a part of our curriculum. We thank the Head of our Computer Department, Prof. (Dr.) Vivek Deshpande for imbibing in us the discipline to go through this project with great determination and efforts. We also thank our project guide, Prof. (Dr.) Ketan Bahulkar for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by him time to time shall carry us a long way in the journey of life on which we are about to embark.

We would also like to express a deep sense of gratitude to all the teachers for their cordial support and guidance, which helped us in completing this task through various stages.

We are obliged to the staff members of our college, for the valuable information provided by them in their respective fields. We are grateful for their cooperation during the period of our project.

ABSTRACT

In today’s busy world, anyone would love a system which will improve the tiresome shopping experience for them. Our system will take the shopping list, maximum budget as input. And will provide the optimised shopping list for the user with product recommendations, recipes the user can try and statistics. As well as, people will be benefited as they won’t have to think about their budget and optimise their shopping list manually which will take a lot of time. The product recommendations will provide the user with more choices helping them to be reminded of some things to buy which they might forget while shopping. And from the list of recipes based on the user’s shopping list, the user can try a new recipe. The user dashboard will provide helpful statistics to the user. Thus, our Intelligent Shopper will help people save their time, money and energy. This report briefly describes the current and planned features of our Intelligent Shopper in a detailed manner.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Retail as a concept has come a long way from the old system of market based on barter, to one stop supermarkets, and are now evolving to a connected Omni-channel retail platform to accommodate behavior of millennial shoppers. And so have the customers steadily moved from an era of lack of choice and payment ambiguity to ample choices and flexible payment terms. The dexterity of customers with access to forums, social media and mobility enabling them to seek out the best deals, with one-click information, has made the customer’s shopping behavior versatile. So, now with personalized mobile and data-driven shopping techniques, the physical shopping experience is converging with the digital experiences to create a revolutionary retail experience based on ease, convenience and excitement.

Retailers have always tried to find ways to help customers find the right product at the right time to make their shopping experience easier, ensuring retailers can provide a seamless and complete shopping experience. For the customers to have this seamless shopping experience across different channels, retailers are looking at ways to combine online, mobile and in-store shopping. In this conquest, new technological solutions are being experimented upon to augment the current shopping experience.

There are numbers of problems exist for shoppers without using any shopping assistant application during their shopping activities. One of the problems is that they found it difficult in locating a store which sells the items they want to buy. When people want to buy a particular product, they often do not have idea where they can purchase the items. Thus, it is troublesome for them when they could not find the items at a store they’ve arrived and causing them to have to travel to another store to find the items. This is very time-consuming and cost-wasting.

Another problem is that shoppers are not able to make price comparison for a same type of product. Normally, shoppers can know the price of a product through catalogue or advertisement, but there are only a limited amount of products displayed to them. Making price comparison is impossible as they do not know the price for the other of the same type of products. The product information on the catalogue and advertisement is also very limited, thus shoppers could not fully understand the products and caused them problems in making choices for their shopping plan.

Shoppers often face problem in estimating budget accurately in their shopping plan, causing them to spend over their budget limit during shopping. This is due to different store might have different prices for a particular product and shoppers do not know about the prices, thus making them difficult in estimating an accurate budget.

A solution can be an Intelligent Shopper which will help people improve their shopping experience.

2. EXISTING SYSTEM STUDY

In the existing systems, shoppers often face problem in estimating budget accurately in their shopping plan, causing them to spend over their budget limit during shopping. This is due to different store might have different prices for a particular product and shoppers do not know about the prices, thus making them difficult in estimating an accurate budget.

Some of them do not provide feature which allow users to see their shopping statistics. Therefore, shoppers could not know what do they buy the most or least according to which they could change their shopping habits. Most of the shoppers like to optimize their shopping list according to the budget. However, they could not manually. Lastly, shoppers would forget some things they wanted to buy while shopping. Hence, the existing systems do not give the shoppers a hassle free, time, energy and money-saving experience.

2.1. EXISTING SYSTEM MODELING

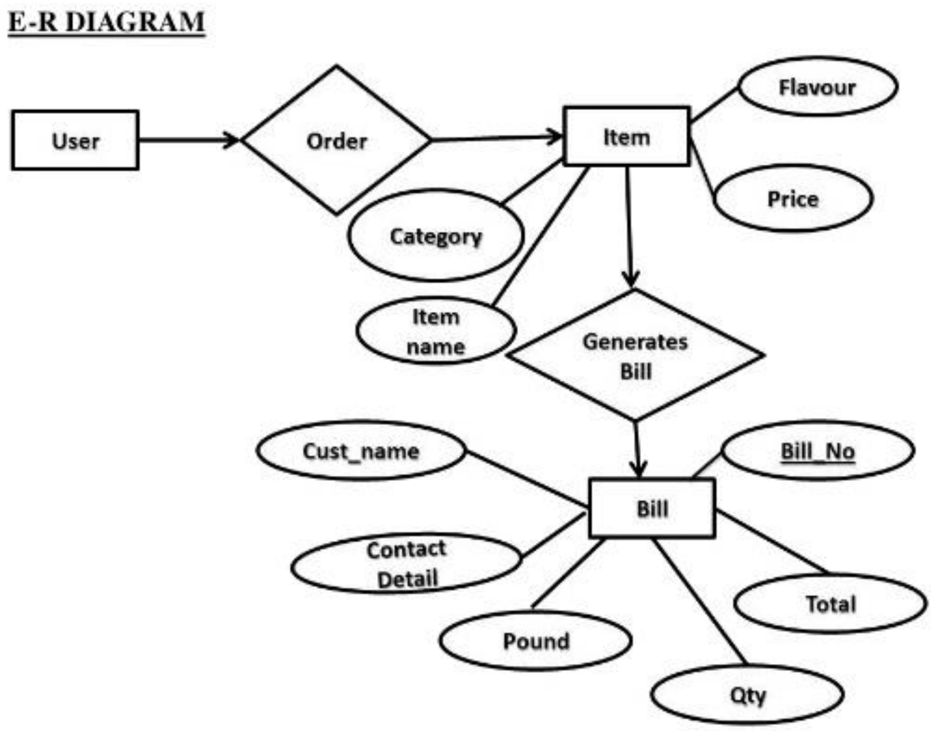


Figure 1: Existing Model's ER Diagram

2.2. DRAWBACKS OF THE EXISTING SYSTEM

As we can observe from the existing system's model, the existing systems do not provide the users to manage their shopping list according to their budget. They also do not provide with product recommendations to help them be reminded of things they need to buy. The existing systems also do not provide any statistical information to the users so that they can improve their shopping habits.

When people want to shop, they must go to the store and look for the items they want to buy. However, it might bring trouble to them if they could not find the items in the store for which they will have to travel to another store to find it. Therefore, with the development of an Intelligent Shopping Assistant, shoppers will no longer face difficulty in obtaining the information and details of the products thus, helping them in making a better shopping plan before going out to shop. They will also get additional facilities such as product recommendations, recipes and dashboard which they did not get from the existing systems.

2.3. PROBLEM DEFINITION

The main objective of our project is:

- ✓ To help the user easily manage his shopping list according to his budget.
- ✓ To predict user's family type by using his shopping lists.
- ✓ To provide product recommendations to the user.
- ✓ To provide new recipes to the users to try.
- ✓ To provide a user dashboard with the statistical information of the user's shopping history.
- ✓ To allow the administrator to generate associations through which he can provide product recommendations to the users efficiently.
- ✓ To provide an admin dashboard which will help the administrator to do product analysis and improve his shop.
- ✓ To provide a helpful interface by which user can easily manage the huge database.

2.4. REQUIREMENTS SPECIFICATION

2.4.1. FUNCTIONAL REQUIREMENTS

- ➢ **DISPLAYING PRODUCTS ACCORDING TO THEIR AVAILABILITY IN THE STORE:**

  Our website will provide the product information to users. It allows sellers to post the product information in their store into the application. This information includes price, shop's name and location. Shoppers have to provide the store name, budget and their shopping list. Hence, the shoppers can know which products are actually available in the store which they have specified.

- **MANAGING SHOPPING LIST ACCORDING TO THE BUDGET:**

  To provide a feature which helps users to make an accurate budget calculation before shopping. Our website allows users to add products they want to buy into a shopping list. The system will then calculate the total price of all the products automatically and display it to users. Therefore, users can avoid the problem of spending beyond budget during shopping.

- **PREDICTING USER'S FAMILY TYPE:**

  As our project name says, Intelligent Shopper, it will be called intelligent only if it does the same. Here, the user has to only input his login details, budget and shopping list and still we can predict the family type of the user. We are making this possible by using SVM (Support Vector Machine) which predicts the family type of the user by analyzing his shopping list. The family type can be categorized into the following four types:

  - 0 → Bachelor Male
  - 1 → Bachelor Female
  - 2 → Couple
  - 3 → Couple with children

  Hence, using the family type, the admin is able to do product analysis and improve his shop based on the frequency of the products being bought by specific family types.

- **PROVIDING PRODUCT RECOMMENDATIONS:**

  In order to make personalized product recommendations, we have used generalized association rules mined from a large set of shopping data. Through the association rules and Apriori's algorithm, we are able to provide fairly accurate product suggestions to the user so that he can be reminded of things he needs to buy.

- **PROVIDING LIST OF RECIPES WHICH THE USER CAN TRY:**

  A common scenario is that a person is doing his or her shopping after work, but is unsure of what to cook. The person might have some left over ham in the fridge, and would like to make use of it. In this scenario, the person could use recipe search feature to look for recipes containing ham. The ingredients can then easily be added to the shopping list if desired. We can detect and recommend recipes based on the items on the shopping list. If it seems from already selected items that a certain dish is to be prepared, the missing ingredients are added to the user's shopping list. If the user has some of the missing ingredient already at home, then he can remove them from the shopping list.

- **USER DASHBOARD:**
  The user dashboard will provide statistical information from the user's shopping history. Using the dashboard, the user can improve his shopping habits.

- **GENERATING ASSOCIATION RULES:**
  This function will allow the administrator to make any changes to the support and confidence parameters while generating the product recommendations. Market Basket Analysis and Apriori's algorithm are used to generate associations and provide fairly accurate product recommendations.

- **ADMIN DASHBOARD:**
  The admin dashboard provides the administrator with useful information with the help of a pie chart and bar graph. The admin selects a product and comes to know statistics about which family type is buying the product most frequently. And according to these details, the admin can make improvements to his shop and progress.

- **MANAGE DATABASE:**
  The admin can manage the database easily using the administration framework provided by Django.

3. PROPOSED SYSTEM

3.1 Proposed Solution

The proposed approach in this project is to develop a web-based application, named Intelligent Shopper. In this application, there are a number of features which are implemented. Firstly, sellers are allowed to add new products on the database with required information. Shoppers can select the store and specify the budget. Then, he can create his shopping list by adding products on the application by choosing product category. The total price will be displayed and the user can manage the shopping list if the total price exceeds the budget specified by him.

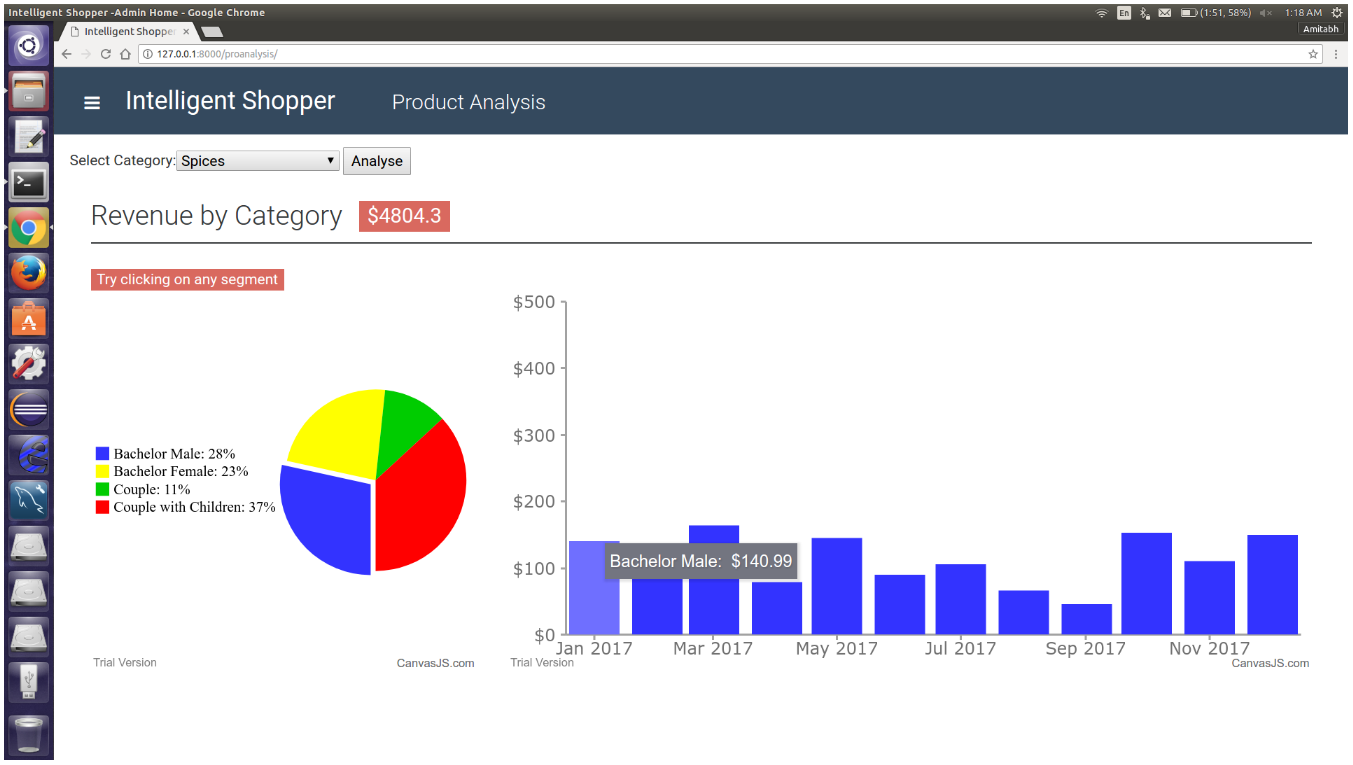
The application will provide product suggestions based on the user's shopping list. Through the product suggestions, the shopper will be reminded of things he wanted to buy in case he might have forgotten. It also provides a list of recipes which the user can try. It predicts the user's family type without taking any detailed inputs from the user. It also provides user dashboard which includes statistical information from the user's shopping history. Using the dashboard, the user can improve his shopping habits.

On the other hand, the application also provides admin with a large set of functionalities. Firstly, the admin can make changes to the association rules based on specific parameters, such as, support and confidence used by the Market Basket Analysis and hence, generate new association rules in case he wants to assess the accuracy of the product suggestions facility he is providing to the users.

The application provides the admin with a dashboard so that he can do product analysis and improve his shop. Product analysis can be done using the statistical information provided in the dashboard which is based on the frequency of the products bought by different family types. The administrator can manage the database easily and efficiently using the administration framework provided by Django.

Hence, our Intelligent Shopper provides a plethora of functions to the users and admin which makes the tiresome shopping experience a happy and satisfying one.

3.2. Working of the System



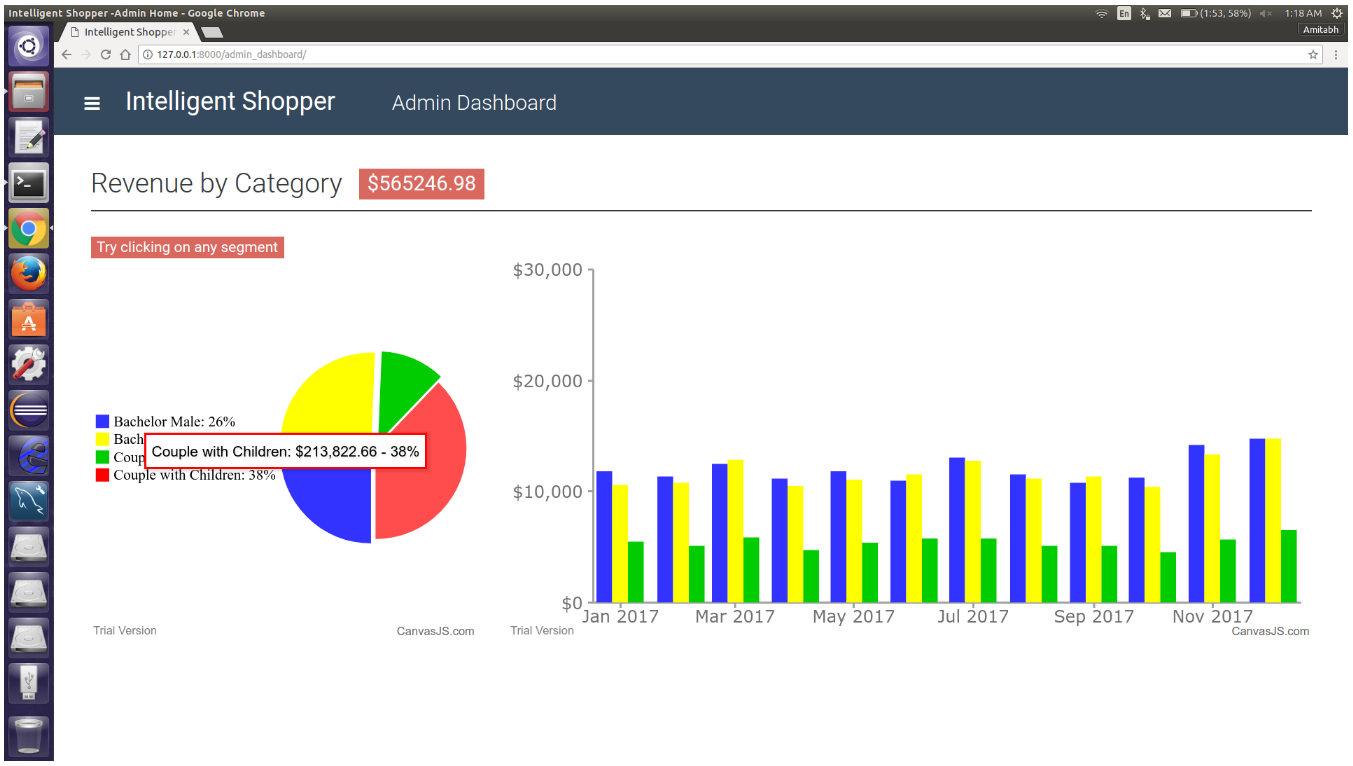
Intelligent Shopper   Product Analysis

Select Category: Spices

Revenue by Category $4804.3

🟦 Bachelor Male: 28%
🟨 Bachelor Female: 23%
🟩 Couple: 11%
🟥 Couple with Children: 37%

Bachelor Male: $140.99

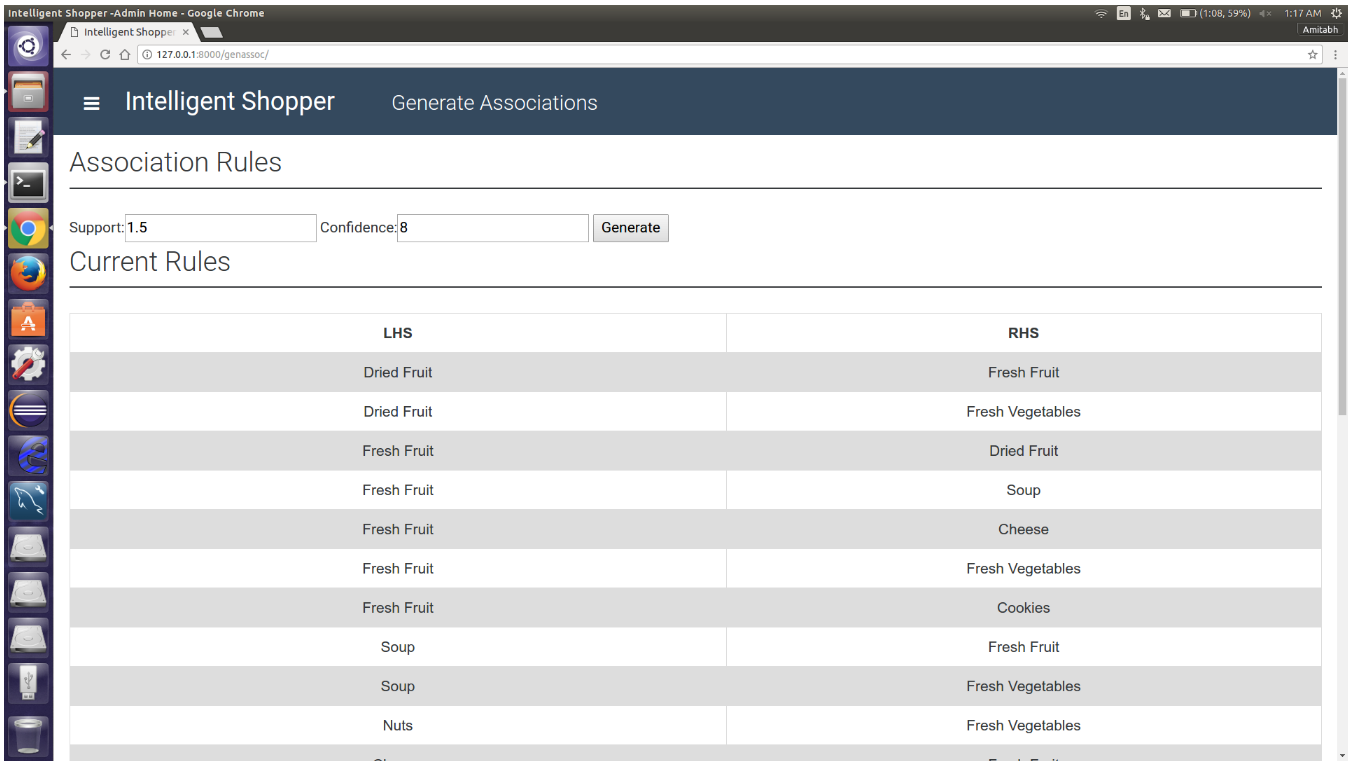


Intelligent Shopper   Admin Dashboard

Revenue by Category $565246.98

🟦 Bachelor Male: 26%
🟨 Bach
🟩 Coup
🟥 Couple with Children: 38%

Couple with Children: $213,822.66 - 38%

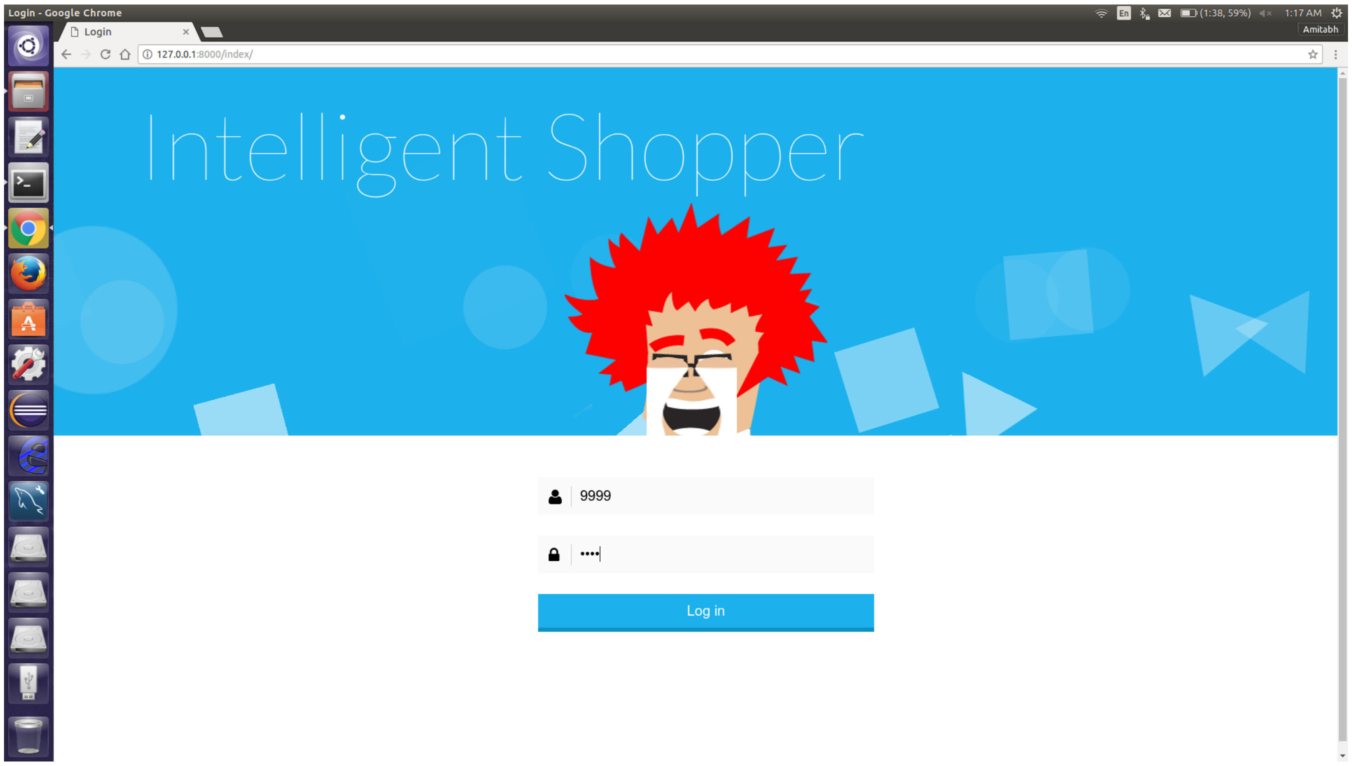


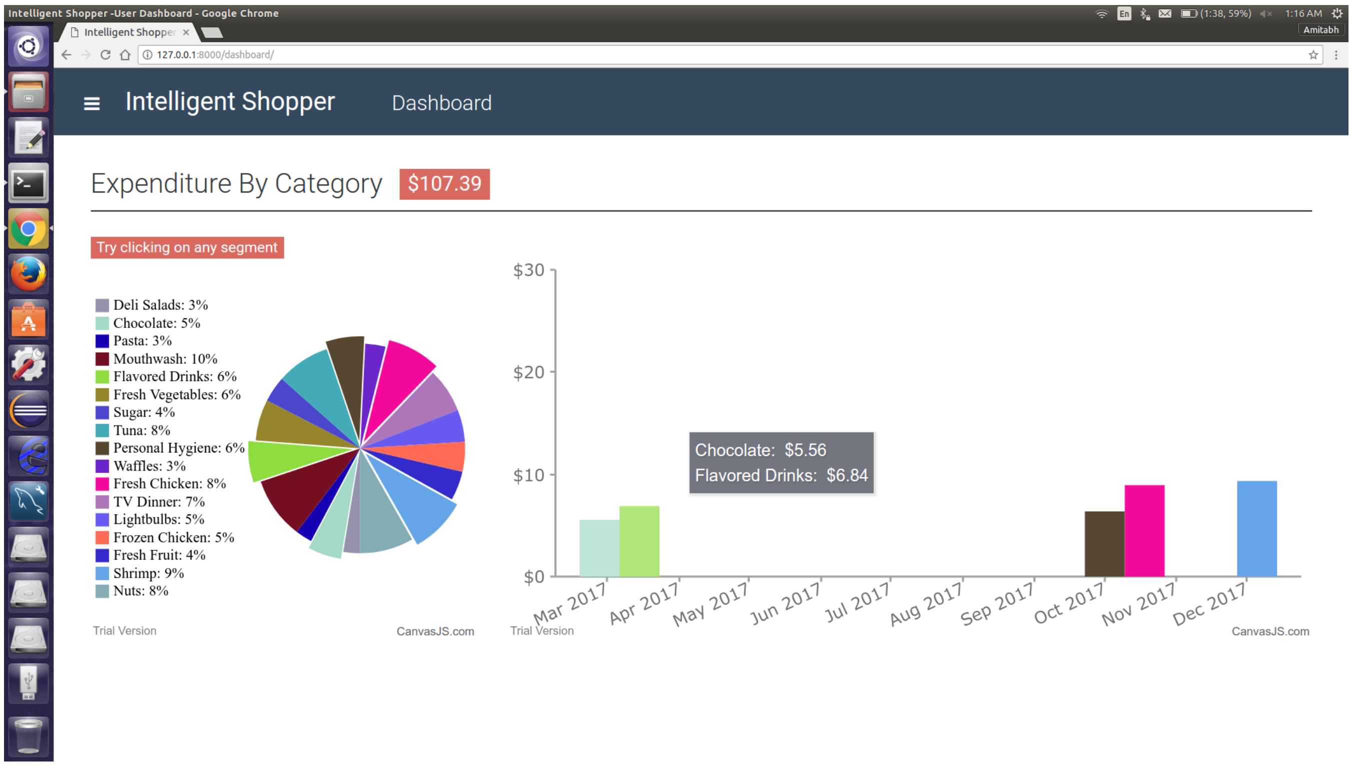
Association Rules

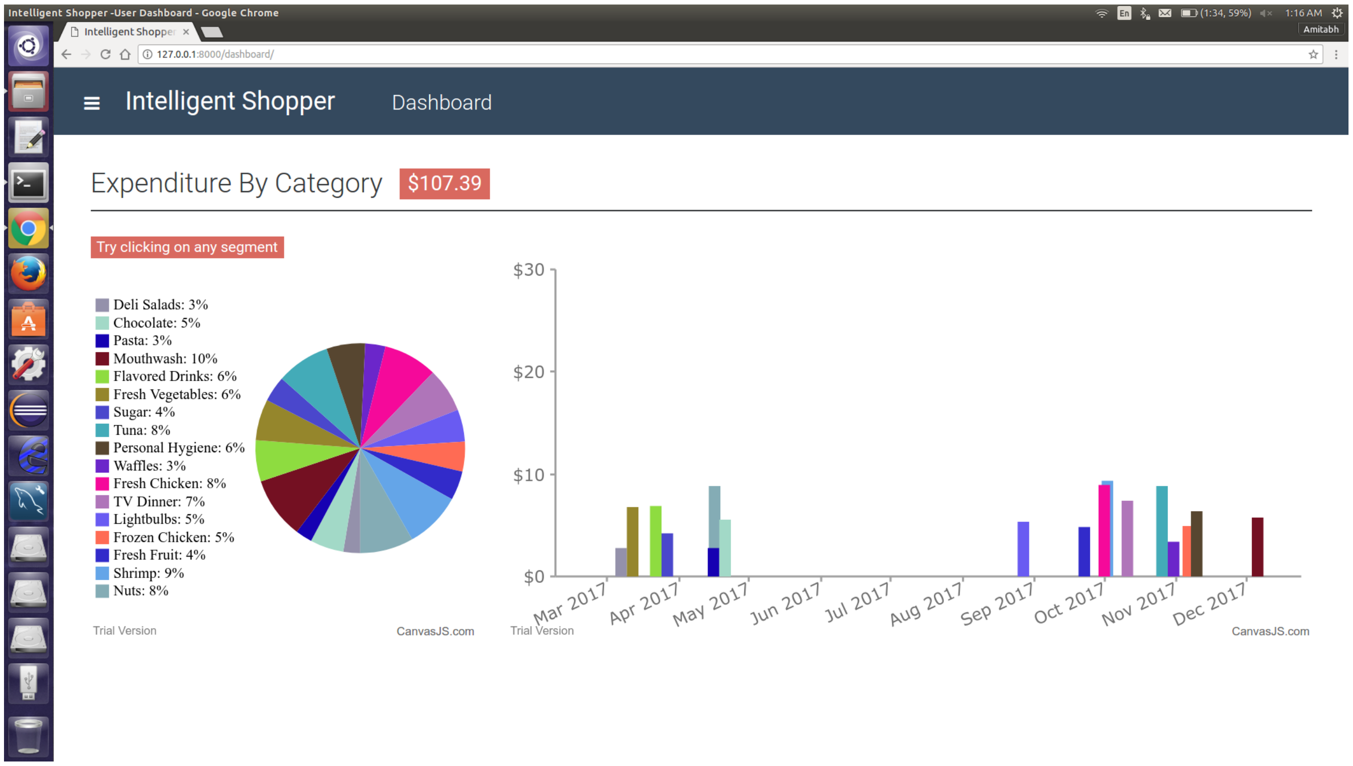
Support: 1.5 Confidence: 8 Generate

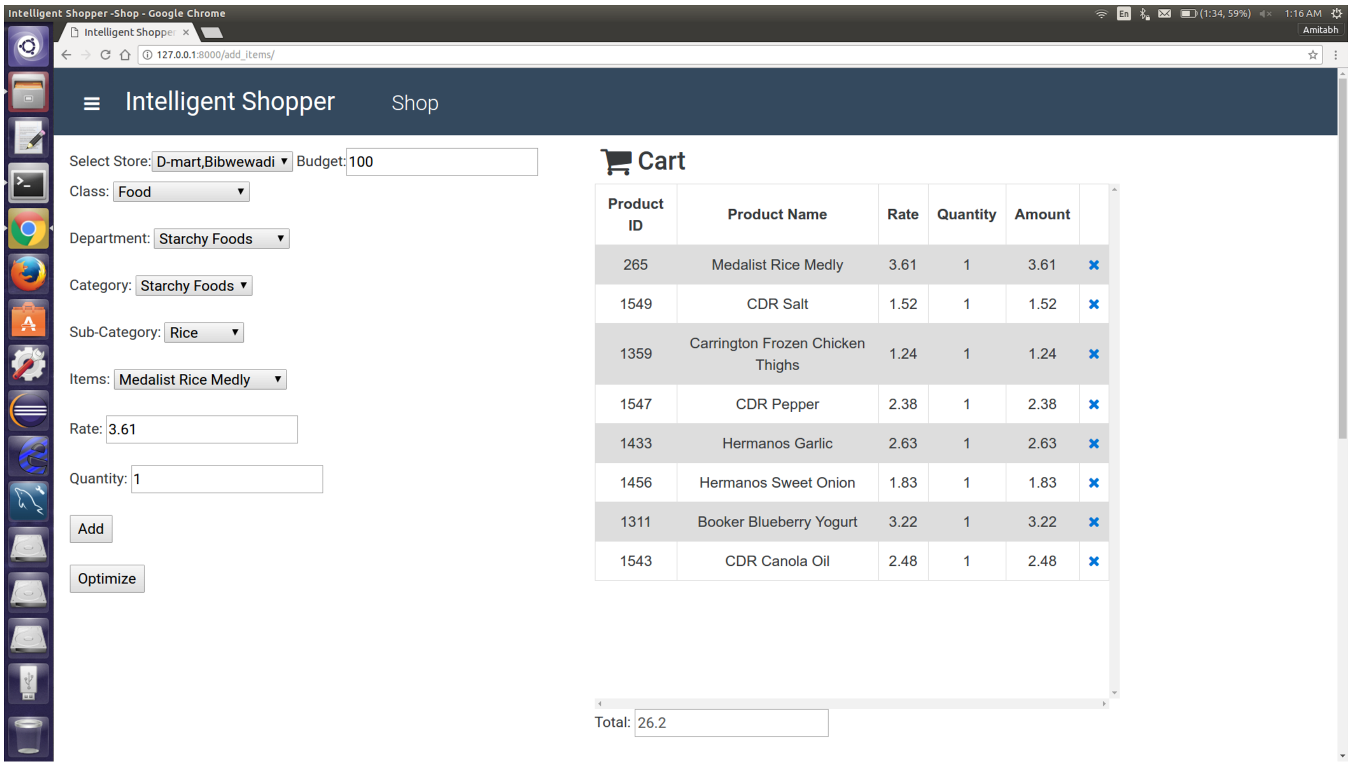
Current Rules

LHS	RHS
Dried Fruit	Fresh Fruit
Dried Fruit	Fresh Vegetables
Fresh Fruit	Dried Fruit
Fresh Fruit	Soup
Fresh Fruit	Cheese
Fresh Fruit	Fresh Vegetables
Fresh Fruit	Cookies
Soup	Fresh Fruit
Soup	Fresh Vegetables
Nuts	Fresh Vegetables




Intelligent Shopper -User Dashboard - Google Chrome
127.0.0.1:8000/dashboard/
Intelligent Shopper
Dashboard
Expenditure By Category
$107.39
Try clicking on any segment
Deli Salads: 3%
Chocolate: 5%
Pasta: 3%
Mouthwash: 10%
Flavored Drinks: 6%
Fresh Vegetables: 6%
Sugar: 4%
Tuna: 8%
Personal Hygiene: 6%
Waffles: 3%
Fresh Chicken: 8%
TV Dinner: 7%
Lightbulbs: 5%
Frozen Chicken: 5%
Fresh Fruit: 4%
Shrimp: 9%
Nuts: 8%
$30
$20
$10
$0
Chocolate: $5.56
Flavored Drinks: $6.84
Mar 2017
Apr 2017
May 2017
Jun 2017
Jul 2017
Aug 2017
Sep 2017
Oct 2017
Nov 2017
Dec 2017
Trial Version
CanvasJS.com


Intelligent Shopper -User Dashboard - Google Chrome
127.0.0.1:8000/dashboard/
Intelligent Shopper
Dashboard
Expenditure By Category
$107.39
Try clicking on any segment
Deli Salads: 3%
Chocolate: 5%
Pasta: 3%
Mouthwash: 10%
Flavored Drinks: 6%
Fresh Vegetables: 6%
Sugar: 4%
Tuna: 8%
Personal Hygiene: 6%
Waffles: 3%
Fresh Chicken: 8%
TV Dinner: 7%
Lightbulbs: 5%
Frozen Chicken: 5%
Fresh Fruit: 4%
Shrimp: 9%
Nuts: 8%
$30
$20
$10
$0
Mar 2017
Apr 2017
May 2017
Jun 2017
Jul 2017
Aug 2017
Sep 2017
Oct 2017
Nov 2017
Dec 2017
Trial Version
CanvasJS.com



Intelligent Shopper Shop

Select Store: D-mart,Bibwewadi Budget: 100

Class: Food

Department: Starchy Foods

Category: Starchy Foods

Sub-Category: Rice

Items: Medalist Rice Medly

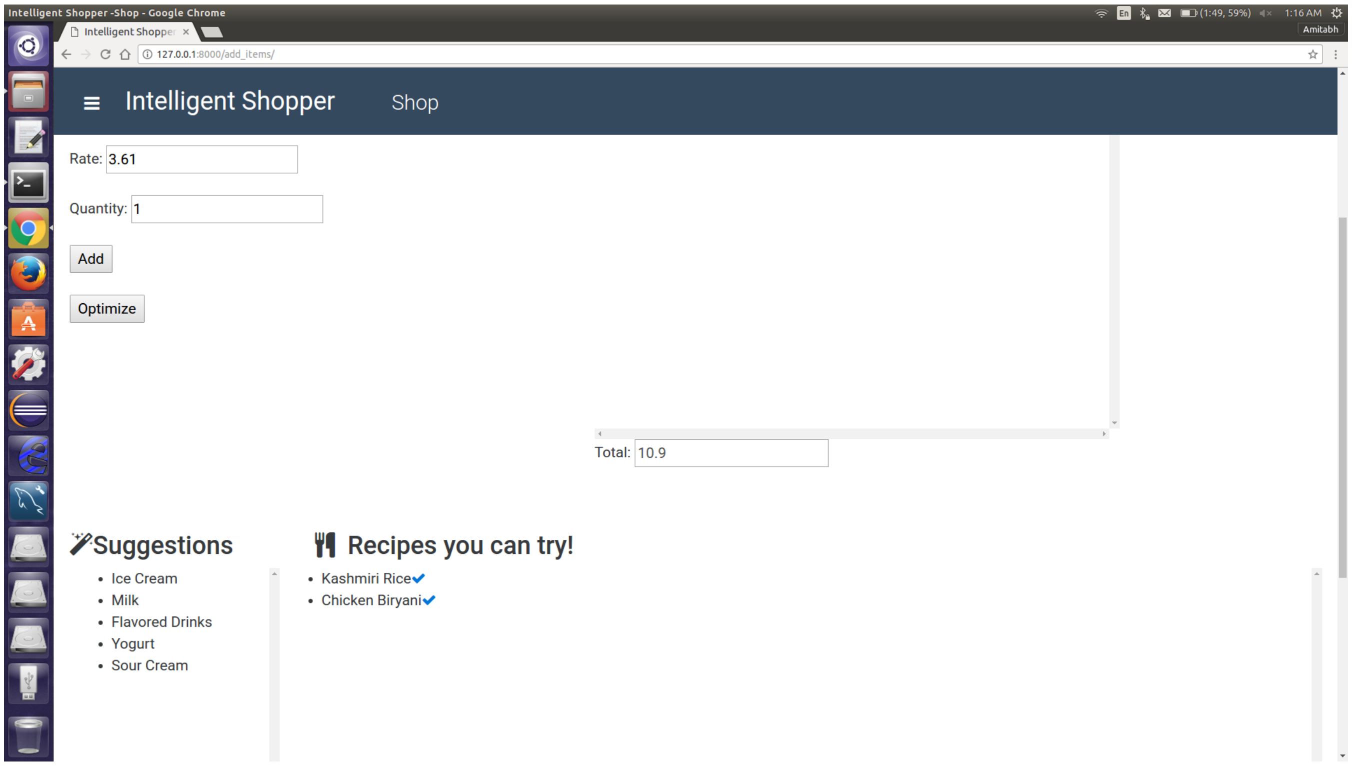
Rate: 3.61

Quantity: 1

Cart

Product ID	Product Name	Rate	Quantity	Amount	
265	Medalist Rice Medly	3.61	1	3.61	✖
1549	CDR Salt	1.52	1	1.52	✖
1359	Carrington Frozen Chicken Thighs	1.24	1	1.24	✖
1547	CDR Pepper	2.38	1	2.38	✖
1433	Hermanos Garlic	2.63	1	2.63	✖
1456	Hermanos Sweet Onion	1.83	1	1.83	✖
1311	Booker Blueberry Yogurt	3.22	1	3.22	✖
1543	CDR Canola Oil	2.48	1	2.48	✖

Total: 26.2



Intelligent Shopper Shop

Rate: 3.61

Quantity: 1

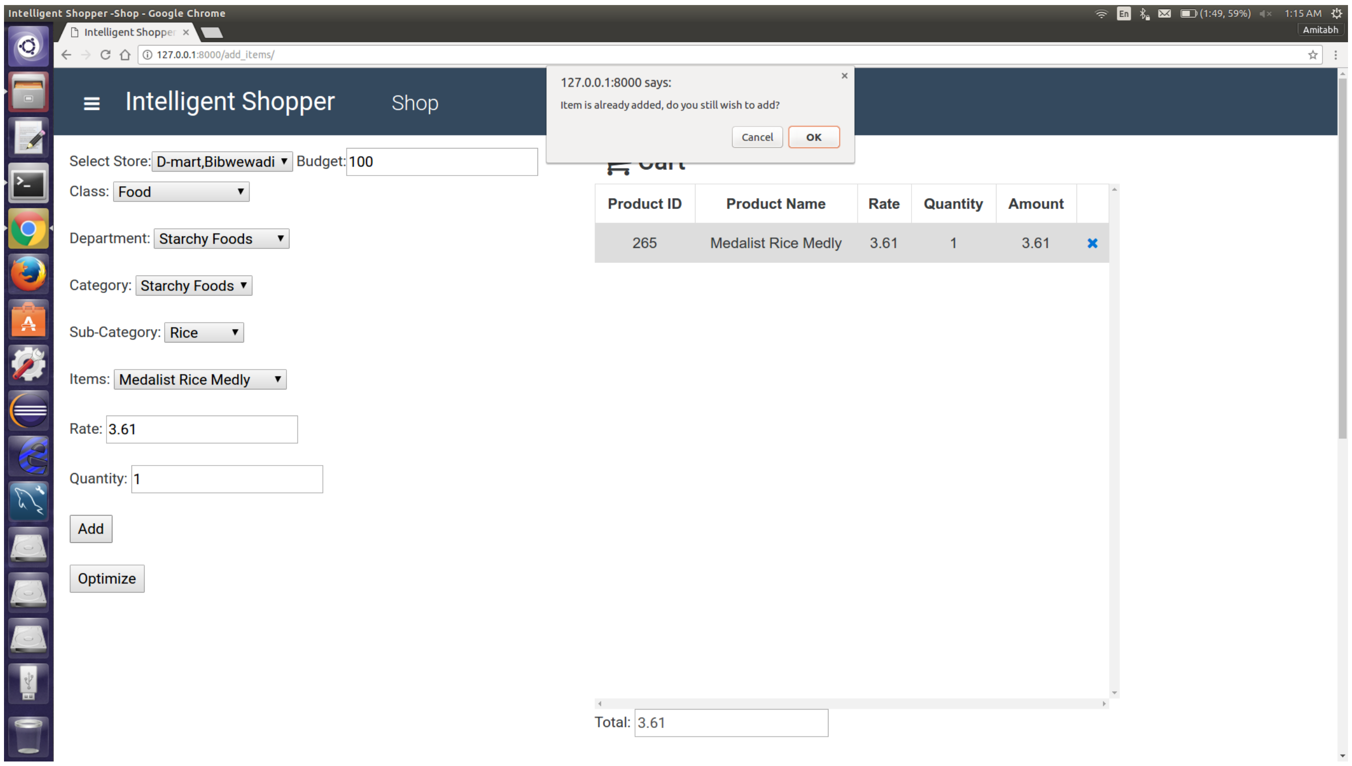
Total: 10.9

Suggestions

- Ice Cream
- Milk
- Flavored Drinks
- Yogurt
- Sour Cream

Recipes you can try!

- Kashmiri Rice✓
- Chicken Biryani✓



Intelligent Shopper -Shop - Google Chrome

127.0.0.1:8000/add_items/

127.0.0.1:8000 says:

Item is already added, do you still wish to add?

Select Store: D-mart,Bibwewadi Budget: 100

Class: Food

Department: Starchy Foods

Category: Starchy Foods

Sub-Category: Rice

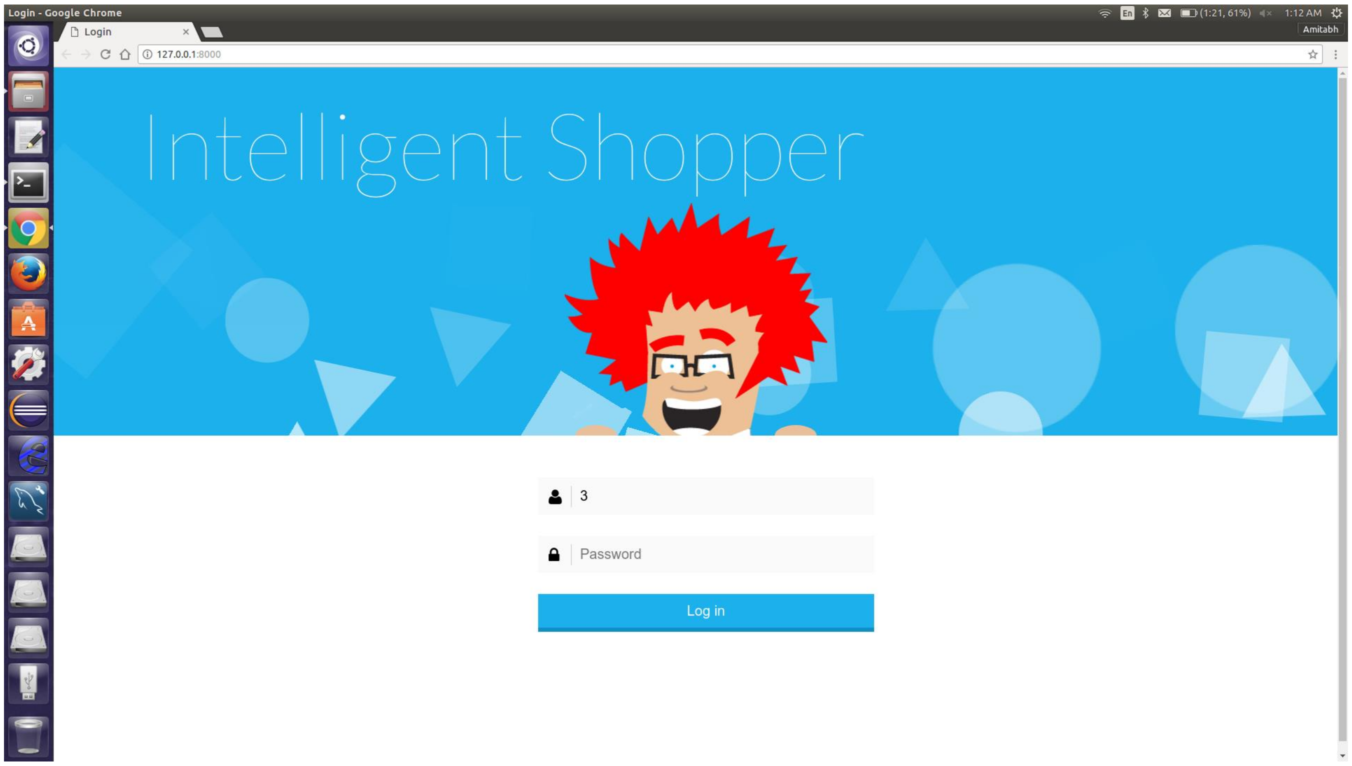
Items: Medalist Rice Medly

Rate: 3.61

Quantity: 1

Product ID	Product Name	Rate	Quantity	Amount
265	Medalist Rice Medly	3.61	1	3.61

Total: 3.61

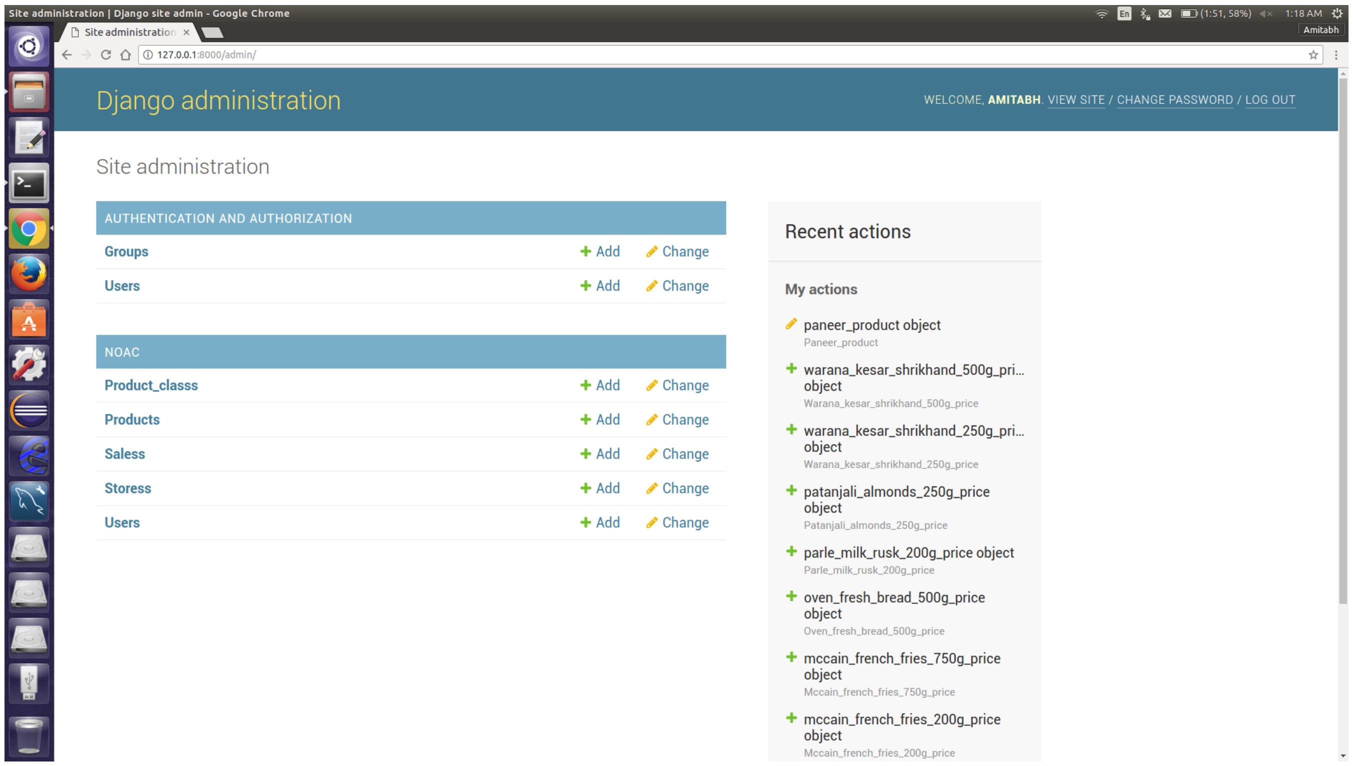


Login - Google Chrome

127.0.0.1:8000

3

Password


Site administration | Django site admin - Google Chrome

127.0.0.1:8000/admin/

Django administration

WELCOME, **AMITABH**. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change

NOAC		
Product_classs	+ Add	Change
Products	+ Add	Change
Saless	+ Add	Change
Storess	+ Add	Change
Users	+ Add	Change

Recent actions

My actions

- paneer_product object
  Paneer_product
- warana_kesar_shrikhand_500g_pri... object
  Warana_kesar_shrikhand_500g_price
- warana_kesar_shrikhand_250g_pri... object
  Warana_kesar_shrikhand_250g_price
- patanjali_almonds_250g_price object
  Patanjali_almonds_250g_price
- parle_milk_rusk_200g_price object
  Parle_milk_rusk_200g_price
- oven_fresh_bread_500g_price object
  Oven_fresh_bread_500g_price
- mccain_french_fries_750g_price object
  Mccain_french_fries_750g_price
- mccain_french_fries_200g_price object
  Mccain_french_fries_200g_price

Figure 2: Working of our System

3.3. Estimation

3.3.1 Hardware

A computer or mobile with a stable connection network.

3.3.2 Software

3.3.2.1 Django Framework

Django is a free and open-source web framework, written in Python, which follows the model-view-template (MVT) architectural pattern. It is maintained by the Django Software Foundation (DSF), an independent organization established as a 501(c) (3) non-profit.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes reusability and "pluggability" of components, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models. Some well-known sites that use Django include the Public Broadcasting

Service Pinterest, Instagram, Mozilla, The Washington Times, Disqus, Bitbucket, and Nextdoor.

3.3.2.1.1 Working of Django

The Model-View-Template (MVT) is slightly different from MVC. In fact the main difference between the two patterns is that Django itself takes care of the Controller part (Software Code that controls the interactions between the Model and View), leaving us with the template. The template is a HTML file mixed with Django Template Language (DTL).

The following diagram illustrates how each of the components of the MVT pattern interacts with each other to serve a user request −

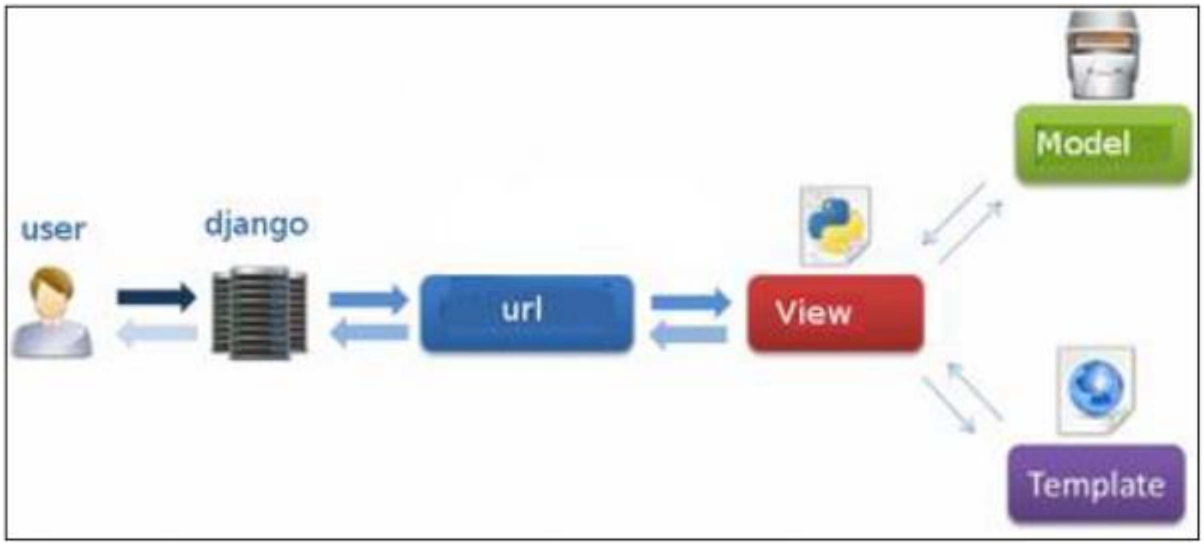


Figure 3: Django's Architecture

The developer provides the Model, the view and the template then just maps it to a URL and Django does the magic to serve it to the user.

Django development environment consists of installing and setting up Python, Django, and a Database System. Since Django deals with web application, it's worth mentioning that you would need a web server setup as well.

Step 1 – Installing Python

Django is written in 100% pure Python code, so you'll need to install Python on your system. Latest Django version requires Python 2.6.5 or higher

If you're on one of the latest Linux or Mac OS X distribution, you probably already have Python installed. You can verify it by typing *python* command at a command prompt. If you see something like this, then Python is installed.

```
$ python
Python 2.7.5 (default, Jun 17 2014, 18:11:42)
[GCC 4.8.2 20140120 (Red Hat 4.8.2-16)] on linux2
```

Otherwise, you can download and install the latest version of Python from the link http://www.python.org/download.

Step 2 - Installing Django

Installing Django is very easy, but the steps required for its installation depends on your operating system. Since Python is a platform-independent language, Django has one package that works everywhere regardless of your operating system.

You can download the latest version of Django from the link http://www.djangoproject.com/download.

UNIX/Linux and Mac OS X Installation

You have two ways of installing Django if you are running Linux or Mac OS system −

- You can use the package manager of your OS, or use easy_install or pip if installed.
- Install it manually using the official archive you downloaded before.

We will cover the second option as the first one depends on your OS distribution. If you have decided to follow the first option, just be careful about the version of Django you are installing.

Let's say you got your archive from the link above, it should be something like Django-x.xx.tar.gz:

Extract and install.

```
$ tar xzvf Django-x.xx.tar.gz
$ cd Django-x.xx
$ sudo python setup.py install
```

You can test your installation by running this command −

```
$ django-admin.py --version
```

If you see the current version of Django printed on the screen, then everything is set.

Note − For some version of Django it will be django-admin the ".py" is removed.

Step 3 – Database Setup

Django supports several major database engines and you can set up any of them based on your comfort.

- MySQL (http://www.mysql.com/)
- PostgreSQL (http://www.postgresql.org/)
- SQLite 3 (http://www.sqlite.org/)
- Oracle (http://www.oracle.com/)
- MongoDb (https://django-mongodb-engine.readthedocs.org)
- GoogleAppEngine Datastore (https://cloud.google.com/appengine/articles/django-nonrel)

Step 4 – Web Server

Django comes with a lightweight web server for developing and testing applications. This server is pre-configured to work with Django, and more importantly, it restarts whenever you modify the code.

However, Django does support Apache and other popular web servers such as Lighttpd.

Now that we have installed Django, let's start using it. In Django, every web app you want to create is called a project; and a project is a sum of applications. An application is a set of code files relying on the MVT pattern. As example let's say we want to build a website, the website is our project and, the forum, news, contact engine are applications. This structure makes it easier to move an application between projects since every application is independent.

Create a Project

Whether you are on Windows or Linux, just get a terminal or a **cmd** prompt and navigate to the place you want your project to be created, then use this code −

```
$ django-admin startproject myproject
```

This will create a "myproject" folder with the following structure −

```
myproject/
   manage.py
   myproject/
      __init__.py
      settings.py
      urls.py
   wsgi.py
```

The Project Structure

The “myproject” folder is just your project container, it actually contains two elements −

- **manage.py** − This file is kind of your project local django-admin for interacting with your project via command line (start the development server, sync db...). To get a full list of command accessible via manage.py you can use the code −

```
$ python manage.py help
```

- **The “myproject” subfolder** − This folder is the actual python package of your project. It contains four files −
    - **__init__.py** − Just for python, treat this folder as package.
    - **settings.py** − As the name indicates, your project settings.
    - **urls.py** − All links of your project and the function to call. A kind of ToC of your project.
    - **wsgi.py** − If you need to deploy your project over WSGI.

Setting Up Your Project

Your project is set up in the subfolder myproject/settings.py. Following are some important options you might need to set −

```
DEBUG = True
```

This option lets you set if your project is in debug mode or not. Debug mode lets you get more information about your project's error. Never set it to ‘True’ for a live project. However, this has to be set to ‘True’ if you want the Django light server to serve static files. Do it only in the development mode.

```
DATABASES = {
  'default': {
    'ENGINE': 'django.db.backends.sqlite3',
    'NAME': 'database.sql',
    'USER': '',
    'PASSWORD': '',
    'HOST': '',
      'PORT': '',
   }
}
```

Database is set in the 'Database' dictionary. The example above is for SQLite engine. As stated earlier, Django also supports −

- MySQL (django.db.backends.mysql)
- PostGreSQL (django.db.backends.postgresql_psycopg2)
- Oracle (django.db.backends.oracle) and NoSQL DB
- MongoDB (django_mongodb_engine)

Before setting any new engine, make sure you have the correct db driver installed.

You can also set others options like: TIME_ZONE, LANGUAGE_CODE, TEMPLATE…

Now that your project is created and configured make sure it's working −

```
$ python manage.py runserver
```

You will get something like the following on running the above code −

```
Validating models...

0 errors found
September 03, 2015 - 11:41:50
Django version 1.6.11, using settings 'myproject.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

A project is a sum of many applications. Every application has an objective and can be reused into another project, like the contact form on a website can be an application, and can be reused for others.

3.3.2.2 Weka

Weka (pronounced to rhyme with Mecca) contains a collection of visualization tools and algorithms for data analysis and predictive modelling, together with graphical user interfaces for easy access to these functions. The original non-Java version of Weka was a Tcl/Tk front-end to (mostly third-party) modelling algorithms implemented in other programming languages, plus data pre-processing utilities in C, and a Makefile-based system for running machine learning experiments. This original version was primarily designed as a tool for analysing data from agricultural domains, but the more recent fully Java-based version (Weka 3), for which development started in 1997, is now used in many different application areas, in particular for educational purposes and research. Advantages of Weka include:

- Free availability under the GNU General Public License.
- Portability, since it is fully implemented in the Java programming language and thus runs on almost any modern computing platform.
- A comprehensive collection of data preprocessing and modeling techniques.
- Ease of use due to its graphical user interfaces.

Weka supports several standard data mining tasks, more specifically, data preprocessing, clustering, classification, regression, visualization, and feature selection. All of Weka's techniques are predicated on the assumption that the data is available as one flat file or relation, where each data point is described by a fixed number of attributes (normally, numeric or nominal attributes, but some other attribute types are also supported). Weka provides access to SQL databases using Java Database Connectivity and can process the result returned by a database query. It is not capable of multi-relational data mining, but there is separate software for converting a collection of linked database tables into a single table that is suitable for processing using Weka. Another important area that is currently not covered by the algorithms included in the Weka distribution is sequence modeling.

3.3.2.3 Python

Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991. An interpreted language, Python has a design philosophy which emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly braces or keywords), and a syntax which allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java. The language provides constructs intended to enable writing clear programs on both a small and large scale.

Python features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object-oriented, imperative, functional programming, and procedural styles. It has a large and comprehensive standard library.

Python interpreters are available for many operating systems, allowing Python code to run on a wide variety of systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations.

3.3.2.4 JQuery

JQuery is a cross-platform JavaScript library designed to simplify the client-side scripting of HTML. It is free, open-source software using the permissive MIT license. Web analysis indicates that it is the most widely deployed JavaScript library by a large margin.

JQuery’s syntax is designed to make it easier to navigate a document, select DOM elements, create animations, handle events, and develop Ajax applications. JQuery also provides capabilities for developers to create plug-ins on top of the JavaScript library. This enables developers to create abstractions for low-level interaction and animation, advanced effects and high-level, themeable widgets. The modular approach to the jQuery library allows the creation of powerful dynamic web pages and Web applications.

The set of jQuery core features—DOM element selections, traversal and manipulation—enabled by its *selector engine* (named "Sizzle" from v1.3), created a new "programming style", fusing algorithms and DOM data structures. This style influenced the architecture of other JavaScript frameworks like YUI v3 and Dojo, later stimulating the creation of the standard *Selectors API*.

Microsoft and Nokia bundle jQuery on their platforms.Microsoft includes it with Visual Studio for use within Microsoft's ASP.NET AJAX and ASP.NET MVC frameworks while Nokia has integrated it into the Web Run-Time widget development platform.

3.3.2.5 CanvasJS

CanvasJS Charts have a simple API and can render across devices including iPhone, iPad, Android, Windows Phone, Desktops, etc. This allows you to create rich dashboards that work across devices without compromising on maintainability or functionality of your web application. Graphs include several good looking themes and are **10x faster** than conventional Flash / SVG based Charting Libraries – resulting in lightweight, beautiful and responsive dashboards. Checkout an overview of our JavaScript Charts.

3.3.2.6 HTML

HTML is the standard mark-up language for creating Web pages.

- HTML stands for Hyper Text Markup Language
- HTML describes the structure of Web pages using markup
- HTML elements are the building blocks of HTML pages
- HTML elements are represented by tags
- HTML tags label pieces of content such as "heading", "paragraph", "table", and so on
- Browsers do not display the HTML tags, but use them to render the content of the page

3.3.2.7 CSS

CSS stands for **C**ascading **S**tyle **S**heets

- CSS describes **how HTML elements are to be displayed on screen, paper, or in other media**
- CSS **saves a lot of work**. It can control the layout of multiple web pages all at once
- External stylesheets are stored in **CSS files.**

3.3.2.8 MySQL

MySQL is the most popular Open Source Relational SQL Database Management System. MySQL is one of the best RDBMS being used for developing various web-based software applications. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company.

3.3.2.9 JSON

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

3.3.2.10 Algorithms Used:

3.3.2.10.1 SVM (Support Vector Machine)

In machine learning, **support vector machines** (**SVMs**, also **support vector networks**) are supervised learning models with associated learning algorithms that analyse data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

When data are not labelled, supervised learning is not possible, and an unsupervised learning approach is required, which attempts to find natural clustering of the data to groups, and then map new data to these formed groups. The clustering algorithm which provides an improvement to the support vector machines is called **support vector**

clustering and is often used in industrial applications either when data are not labelled or when only some data are labelled as a pre-processing for a classification pass.

3.3.2.10.2 Market Basket Analysis

Market basket analysis might tell a retailer that customers often purchase shampoo and conditioner together, so putting both items on promotion at the same time would not create a significant increase in revenue, while a promotion involving just one of the items would likely drive sales of the other.

Market basket analysis may provide the retailer with information to understand the purchase behaviour of a buyer. This information will enable the retailer to understand the buyer's needs and rewrite the store's layout accordingly, develop cross-promotional programs, or even capture new buyers (much like the cross-selling concept). An apocryphal early illustrative example for this was when one super market chain discovered in its analysis that male customers that bought diapers often bought beer as well, have put the diapers close to beer coolers, and their sales increased dramatically. Although this urban legend is only an example that professors use to illustrate the concept to students, the explanation of this imaginary phenomenon might be that fathers that are sent out to buy diapers often buy a beer as well, as a reward. This kind of analysis is supposedly an example of the use of data mining. A widely used example of cross selling on the web with market basket analysis is Amazon.com's use of "customers who bought book A also bought book B", e.g. "People who read History of Portugal were also interested in Naval History".

Market basket analysis can be used to divide customers into groups. A company could look at what other items people purchase along with eggs, and classify them as baking a cake (if they are buying eggs along with flour and sugar) or making omelettes (if they are buying eggs along with bacon and cheese). This identification could then be used to drive other programs. Similarly, it can be used to divide products into natural groups. A company could look at what products are most frequently sold together and align their category management around these cliques.

3.3.2.10.2.1 Association Rules and Apriori's Algorithm

By the following problem, we will understand Association Rules and Apriori's algorithm.

The Problem

When we go grocery shopping, we often have a standard list of things to buy. Each shopper has a distinctive list, depending on one's needs and preferences. A housewife might buy healthy ingredients for a family dinner, while a bachelor might buy beer and chips. Understanding these buying patterns can help to increase sales in several ways. If there is a pair of items, X and Y that are frequently bought together:

Product placement in Tesco, UK.

- Both X and Y can be placed on the same shelf, so that buyers of one item would be prompted to buy the other.
- Promotional discounts could be applied to just one out of the two items.
- Advertisements on X could be targeted at buyers who purchase Y.
- X and Y could be combined into a new product, such as having Y in flavors of X.

While we may know that certain items are frequently bought together, the question is, how do we uncover these associations?

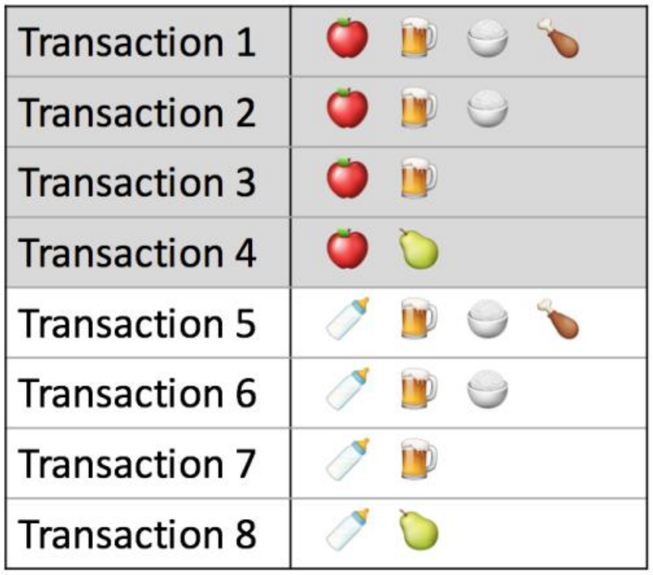
Besides increasing sales profits, association rules can also be used in other fields. In medical diagnosis for instance, understanding which symptoms tend to co-morbid can help to improve patient care and medicine prescription.

Definition

Association rules analysis is a technique to uncover how items are associated to each other. There are three common ways to measure association.

Measure 1: Support. This says how popular an item set is, as measured by the proportion of transactions in which an item set appears. In Table 1 below, the support of {apple} is 4 out of 8, or 50%. Item sets can also contain multiple items. For instance, the support of {apple, beer, rice} is 2 out of 8, or 25%.

Support {🍎} = $\frac{4}{8}$



Transaction 1	🍎 🍺 🍚 🍗
Transaction 2	🍎 🍺 🍚
Transaction 3	🍎 🍺
Transaction 4	🍎 🍐
Transaction 5	🍼 🍺 🍚 🍗
Transaction 6	🍼 🍺 🍚
Transaction 7	🍼 🍺
Transaction 8	🍼 🍐

Table 1. Example Transactions

If you discover that sales of items beyond a certain proportion tend to have a significant impact on your profits, you might consider using that proportion as your *support threshold*. You may then identify item sets with support values above this threshold as significant item sets.

Measure 2: Confidence. This says how likely item Y is purchased when item X is purchased, expressed as {X -> Y}. This is measured by the proportion of transactions with item X, in which item Y also appears. In Table 1, the confidence of {apple -> beer} is 3 out of 4, or 75%.

Confidence {🍎 → 🍺} = $\frac{\text{Support \{🍎, 🍺\}}}{\text{Support \{🍎\}}}$

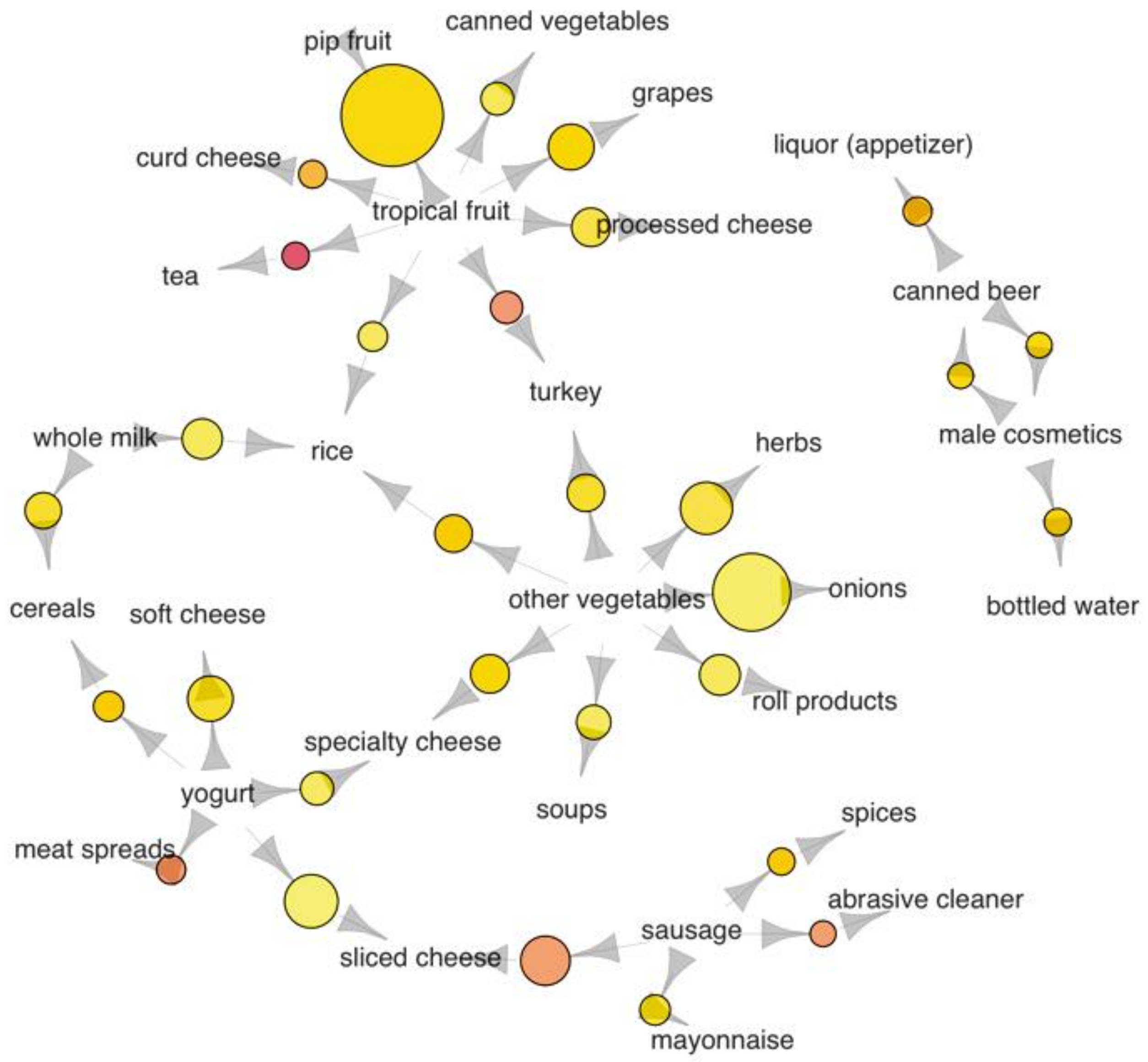
One drawback of the confidence measure is that it might misrepresent the importance of an association. This is because it only accounts for how popular apples are, but not beers. If beers are also very popular in general, there will be a higher chance that a transaction containing apples will also contain beers, thus inflating the confidence measure. To account for the base popularity of both constituent items, we use a third measure called lift.

Measure 3: Lift. This says how likely item Y is purchased when item X is purchased, while controlling for how popular item Y is. In Table 1, the lift of {apple -> beer} is 1, which implies no association between items. A lift value greater than 1 means that item Y is *likely* to be bought if item X is bought, while a value less than 1 means that item Y is *unlikely* to be bought if item X is bought.

$$\text{Lift}\ \{🍎 \rightarrow 🍺\} = \frac{\text{Support}\ \{🍎, 🍺\}}{\text{Support}\ \{🍎\} \times \text{Support}\ \{🍺\}}$$

An Illustration

We use a dataset on grocery transactions from the *rules* R library. It contains actual transactions at a grocery outlet over 30 days. The network graph below shows associations between selected items. Larger circles imply higher support, while red circles imply higher lift:

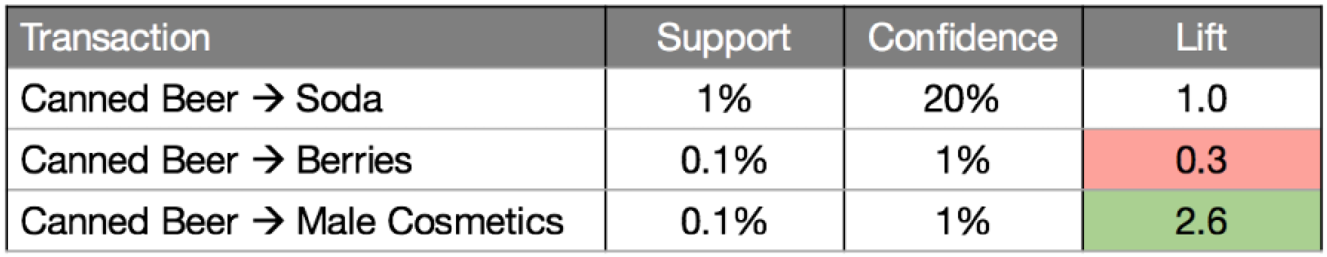


Associations between selected items. Visualized using the rules namely R library.

Several purchase patterns can be observed. For example:

- The most popular transaction was of pip and tropical fruits
- Another popular transaction was of onions and other vegetables
- If someone buys meat spreads, he is likely to have bought yogurt as well
- Relatively many people buy sausage along with sliced cheese
- If someone buys tea, he is likely to have bought fruit as well, possibly inspiring the production of fruit-flavored tea

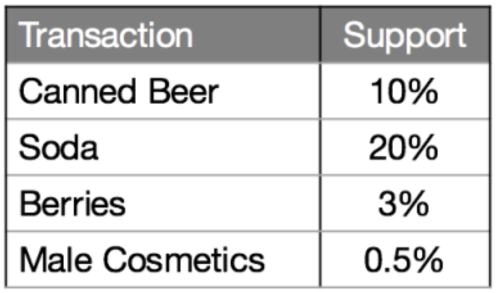
Recall that one drawback of the confidence measure is that it tends to misrepresent the importance of an association. To demonstrate this, we go back to the main dataset to pick 3 association rules containing beer:



Transaction	Support	Confidence	Lift
Canned Beer → Soda	1%	20%	1.0
Canned Beer → Berries	0.1%	1%	0.3
Canned Beer → Male Cosmetics	0.1%	1%	2.6

Table 2. Association measures for beer-related rules

The {beer -> soda} rule has the highest confidence at 20%. However, both beer and soda appear frequently across all transactions (see Table 3), so their association could simply be a fluke. This is confirmed by the lift value of {beer -> soda}, which is 1, implying no association between beer and soda.



Transaction	Support
Canned Beer	10%
Soda	20%
Berries	3%
Male Cosmetics	0.5%

Table 3. Support of individual items

On the other hand, the {beer -> male cosmetics} rule has a low confidence, due to few purchases of male cosmetics in general. However, whenever someone does buy male cosmetics, he is very likely to buy beer as well, as inferred from a high lift value of 2.6. The converse is true for {beer -> berries}. With a lift value below 1, we may conclude that if someone buys berries, he would likely be averse to beer.

It is easy to calculate the popularity of a single item set, like {beer, soda}. However, a business owner would not typically ask about individual item sets. Rather, the owner would be more interested in having a *complete* list of popular item sets. To get this list, one needs to calculate the support values for every possible configuration of items, and then shortlist the item sets that meet the minimum support threshold.

In a store with just 10 items, the total number of possible configurations to examine would be a whopping 1023. This number increases exponentially in a store with hundreds of items.

Is there a way to reduce the number of item configurations to consider?

Apriori Algorithm

The *apriori principle* can reduce the number of item sets we need to examine. Put simply, the apriori principle states that if an item set is infrequent, then all its subsets must also be infrequent. This means that if {beer} was found to be infrequent, we can expect {beer, pizza} to be equally or even more infrequent. So in consolidating the list of popular item sets, we need not consider {beer, pizza}, nor any other item set configuration that contains beer.

Finding item sets with high support

Using the apriori principle, the number of item sets that have to be examined can be pruned, and the list of popular item sets can be obtained in these steps:

Step 0. Start with item sets containing just a single item, such as {apple} and {pear}.
Step 1. Determine the support for item sets. Keep the item sets that meet your minimum support threshold, and remove item sets that do not.
Step 2. Using the item sets you have kept from Step 1, generate all the possible item set configurations.
Step 3. Repeat Steps 1 & 2 until there are no more new item sets.

{apple} was determine to have low support, hence it was removed and all other item set configurations that contain apple need not be considered. This reduced the number of item sets to consider by more than half.

Note that the support threshold that you pick in Step 1 could be based on formal analysis or past experience. If you discover that sales of items beyond a certain proportion tend to have a significant impact on your profits, you might consider using that proportion as your support threshold.

Finding item rules with high confidence or lift

We have seen how the apriori algorithm can be used to identify item sets with high support. The same principle can also be used to identify item associations with high confidence or lift. Finding rules with high confidence or lift is less computationally taxing once high-support item sets have been identified, because confidence and lift values are calculated using support values.

Take for example the task of finding high-confidence rules. If the rule

{beer, chips -> apple}

has low confidence, all other rules with the same constituent items and with apple on the right hand side would have low confidence too. Specifically, the rules

{beer -> apple, chips}
{chips -> apple, beer}

would have low confidence as well. As before, lower level candidate item rules can be pruned using the apriori algorithm, so that fewer candidate rules need to be examined.

4. CONCLUSION

This project is encouraging shoppers to have a concept and idea of managing a smart shopping plan. In fact, most shoppers often do not have a shopping plan before going out for shopping because they do not have the actual information for the products they wanted to buy. Thus, they often face the problem of spending more than expected budget during shopping because they are not able to make estimation on the budget. This project makes a huge contribution in providing them the product information, helping them in organizing their shopping plan as well as controlling their budget.

It will also help the users to be reminded of things they wanted to buy from the product recommendations. So, they don't need to worry about remembering the things they want to buy. The user also gets different recipe suggestions according to his shopping list which helps him in exploring new products which he did not know were good.

As our website also provides a dashboard, the user and admin both get useful information. User gets to know his shopping habits and hence, can improve on that whereas the admin can perform product analysis and progress his shop. The application provides admin a helpful interface to manage the database as well as to alter the association rules for product recommendation facility.

Hence, the tiresome and time-consuming shopping experience can be optimized using our shopping assistant, "The Intelligent Shopper". It will give the users and admin a time, money and energy-saving experience.

5. FUTURE WORK

There is lot of future work in our project. It can be stated as follows:

- **Improving the SVM model file** by removing irrelevant noise.
- **Determining different patterns** observed by our system based on demographics, user profile.
- **Providing health suggestions** for user based on his shopping statistics
- **In-store product locator:** Locate products easily in a hassle-free way.
- **Indoor Navigator:** Integration with indoor navigation for the user who is led to the product
- **Service Related Feedback:** Customer feedback for a particular service or product or suggestions
- **Location Triggered Advertisements:** Our plan is to integrate the product recommendations with advertisements. First of all, matching the recommendations against a database of current product offers makes it possible to rank the advertisements based on how interesting they are to the user. Secondly, the location of the user can be used to trigger the advertisements when the user is near the corresponding products.

6. REFERENCES

1. Adel Alshamrani and Abdullah Bahattab. 2015. A Comparison Between Three SDLC Models Waterfall Model Spiral Model and Incremental/Iterative Model. [ONLINE] Available at: http://www.academia.edu/10793943/A_Comparison_Between_Three_SDLC_Models _Waterfall_Model_Spiral_Model_and_Incremental_Iterative_Model. [Accessed 10 July 15].

2. Bhattacharya, S., Floréen, P., Forsblom, A., Hemminki, S., Myllymäki, P., Nurmi, P., Pulkkinen, T., and Salovaara, A., 2012. Ma$$iv€ - An Intelligent Mobile Grocery Assistant. 8th International Conference on Intelligent Environments Guanajuato. no. 8, pp. 165-171.

3. Davis, Z., Hu, M., Prasad, S., Schuricht, M., Mellier-Smith P., and Moser, L., 2006. A Personal Handheld Multi-Modal Shopping Assistant. Proceedings of International Conference on Networking and Services, pp. 117-125.

4. Mohamed Sami. 2012. Software Development Life Cycle Models and Methodologies. [ONLINE] Available at: https://melsatar.wordpress.com/2012/03/15/softwaredevelopment-life-cycle-models-and-methodologies/. [Accessed 15 July 15].

5. Nair, P., Helal, A. & Shekar, S., 2003. iGrocer- A Ubiquitous and Pervasive Smart Grocery Shopping System. pp.1-7.

6. https://www.tutorialspoint.com/django/django_apps_life_cycle.htm

7. http://www.kdnuggets.com/2016/04/association-rules-apriori-algorithm-tutorial.html

8. http://scikit-learn.org/stable/supervised_learning.html#supervised-learning