**Mini Project: Business Intelligence**

Made by:

Sakshi Khose (501727)

Rutuja Kitukale (501728)

Varad More (501733)

**Problem Statement:**

In this fast-changing digital world, there is a huge competition among different businesses. The sellers who are able to tell the customers what they might want to buy always sustain. it not only improves your sales but also the customer experience. On the other hand, if you are able to predict the next purchase, the customer might always come back to your store.

Our Project aims at developing a system that would be able to predict a customer’s next purchase based on previous yearly sales data of a bakery.

**Data Required for Project:**

We have used the “Bread Basket” Dataset which contains data from a bakery located in Edinburg. The dataset has 21293 entries, over 6000 transactions and 4 columns:

* **Date:** Categorical variable that tells us the date of the transactions (YYYY-MM-DD format).
* **Time:** Categorical variable that tells us the time of the transactions (HH:MM:SS format). Transaction: Quantitative variable that allows us to differentiate the transactions. The rows that share the same value in this field belong to the same transaction, that's why the data set has fewer transactions than observations.
* **Item:** Categorical variable with the products.

**Algorithm Used:**

* **Introduction**

Apriori algorithm was the first algorithm that was proposed for frequent itemset mining. This algorithm uses two steps “join” and “prune” to reduce the search space. It is an iterative approach to discover the most frequent itemsets. The steps followed in the Apriori Algorithm of data mining are:

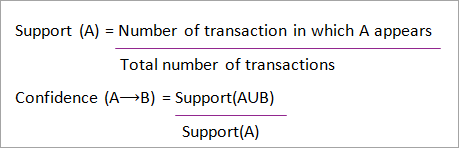
* Join Step: This step generates (K+1) itemset from K-itemsets by joining each item with itself.
* Prune Step: This step scans the count of each item in the database. If the candidate item does not meet minimum support, then it is regarded as infrequent and thus it is removed. This step is performed to reduce the size of the candidate itemsets.
* **Terminologies**

1. **ItemSet**

A set of items together is called an itemset. If any itemset has k-items it is called a k-itemset. An itemset consists of two or more items. An itemset that occurs frequently is called a frequent itemset. A set of items is called frequent if it satisfies a minimum threshold value for support and confidence.

1. **Support and Confidence**

Support shows transactions with items purchased together in a single transaction. Confidence shows transactions where the items are purchased one after the other. Support and confidence can be defined as:

****

1. **Association Rules**

**Association Rule Mining can be defined as:**

“Let I= { …} be a set of ‘n’ binary attributes called items. Let D= { ….} be a set of transactions called databases. Each transaction in D has a unique transaction ID and contains a subset of the items in I. A rule is defined as an implication of form X->Y . The set of items X and Y are called antecedent and consequent of the rule respectively.”

* **Algorithm/Working:** Apriori algorithm is a sequence of steps to be followed to find the most frequent itemset in the given database. This data mining technique follows the join and the prune steps iteratively until the most frequent itemset is achieved. A minimum support threshold is given in the problem or it is assumed by the user.

1. In the first iteration of the algorithm, each item is taken as a 1-itemsets candidate. The algorithm will count the occurrences of each item.
2. Let there be some minimum support, min\_sup. The set of 1 – itemsets whose occurrence is satisfying the min sup are determined. Only those candidates which count more than or equal to min\_sup, are taken ahead for the next iteration and the others are pruned.
3. Next, 2-itemset frequent items with min\_sup are discovered. For this in the join step, the 2-itemset is generated by forming a group of 2 by combining items with itself.
4. The 2-itemset candidates are pruned using a min-sup threshold value. Now the table will have 2 –itemsets with min-sup only.
5. The next iteration will form 3 –itemsets using join and prune step. This iteration will follow anti monotone property where the subsets of 3-itemsets, that is the 2 –itemset subsets of each group fall in min\_sup. If all 2-itemset subsets are frequent then the superset will be frequent otherwise it is pruned.
6. The next step will follow by making a 4-itemset by joining 3-itemset with itself and pruning if its subset does not meet the min\_sup criteria. The algorithm is stopped when the most frequent itemset is achieved.

**Implementation:**

1. **Python code embedded with Flask**

from flask import Flask, request

import matplotlib.pyplot as plt

from matplotlib.backends.backend\_agg import FigureCanvasAgg

from matplotlib.backends.backend\_svg import FigureCanvasSVG

from flask import render\_template

import pandas as pd

import numpy as np

import warnings

from matplotlib.figure import Figure

from werkzeug import secure\_filename

app = Flask(\_\_name\_\_)

@app.route('/')

def hello\_world():

return 'Hello, World!'

#Creating Route - For uploading Dataset

@app.route('/upload')

def upload\_files():

print("##")

return render\_template('uploader.html')

#Creating Route - For Accepting Dataset and producing Visualizations

@app.route('/final', methods = ['GET', 'POST'])

def bil\_aprori():

if request.method == 'POST':

f = request.files['file']

# f.save(secure\_filename(f.filename))

print(f)

app.logger.info("File Received")

else:

return 'Error in Upload'

#Importing Dataset & pre-processing

warnings.filterwarnings('ignore')

df = pd.read\_csv(f)

print ("Dataset Import Success")

df['Item']=df['Item'].str.lower()

#Ignoring blanks in data

x=df['Item']== 'none'

print(x.value\_counts())

df=df.drop(df[df.Item == 'none'].index)

#Extracting Unique tuples

len(df['Item'].unique())

df\_for\_top10\_Items=df['Item'].value\_counts().head(10)

Item\_array= np.arange(len(df\_for\_top10\_Items))

#Plotting Visualization 1 - Maximum sold product

import matplotlib.pyplot as plt

fig, ax=plt.subplots(figsize=(16,7)) df['Item'].value\_counts().sort\_values(ascending=False).head(20).plot.bar(width=0.5,edgecolor='k',align='center',linewidth=1)

plt.xlabel('Food Item',fontsize=20)

plt.ylabel('Number of transactions',fontsize=17)

ax.tick\_params(labelsize=20)

plt.title('20 Most Sold Items',fontsize=20)

# plt.grid()

plt.savefig('static/new\_plot1.png')

plt.clf()

plt.cla()

plt.close()

######################################################################

#Plotting Visualization 2 - Day wise sales

df['Date'] = pd.to\_datetime(df['Date'])

df['Time'] = pd.to\_datetime(df['Time'],format= '%H:%M:%S' ).dt.hour

df['day\_of\_week'] = df['Date'].dt.weekday

d=df.loc[:,'Date']

weekday\_names=[ 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']

Weekday\_number=[0,1,2,3,4,5,6]

week\_df = d.groupby(d.dt.weekday).count().reindex(Weekday\_number)

Item\_array\_week= np.arange(len(week\_df))

plt.figure(figsize=(15,5))

my\_colors = 'rk'

plt.bar(Item\_array\_week,week\_df, color=my\_colors)

plt.xticks(Item\_array\_week,weekday\_names)

plt.title('Number of Transactions made based on Weekdays')

#plt.show()

plt.savefig('static/new\_plot2.png')

plt.clf()

plt.cla()

plt.close()

#####################################################################

#Plotting Visualization 3 - Daily sales distributed per time basis

dt=df.loc[:,'Time']

Hour\_names=[ 7, 8, 9,10,11,12,13,14,15,16,17,18,19,20,21,22,23]

time\_df=dt.groupby(dt).count().reindex(Hour\_names)

Item\_array\_hour= np.arange(len(time\_df))

plt.figure(figsize=(15,5))

my\_colors = 'rb'

plt.bar(Item\_array\_hour,time\_df, color=my\_colors)

plt.xticks(Item\_array\_hour,Hour\_names)

plt.title('Number of Transactions made based on Hours')

#plt.show()

plt.savefig('static/new\_plot3.png')

plt.clf()

plt.cla()

plt.close()

##############################################################################

#Apriori Implementation

from mlxtend.frequent\_patterns import apriori

from mlxtend.frequent\_patterns import association\_rules hot\_encoded\_df=df.groupby(['Transaction','Item'])['Item'].count().unstack().reset\_index().fillna(0).set\_index('Transaction')

def encode\_units(x):

if x <= 0:

return 0

if x >= 1:

return 1

hot\_encoded\_df = hot\_encoded\_df.applymap(encode\_units)

frequent\_itemsets = apriori(hot\_encoded\_df, min\_support=0.01, use\_colnames=True)

rules = association\_rules(frequent\_itemsets, metric="lift", min\_threshold=1)

rules.head()

rules[ (rules['lift'] >= 1) &

(rules['confidence'] >= 0.5) ] # Considering support 0.01 and confidence 50%

###############################################################################

#Plotting Visualization 4 - Association rules scatter plot

support=rules['support'].values

confidence=rules['confidence'].values

for i in range (len(support)):

support[i] = support[i]

confidence[i] = confidence[i]

plt.plot()

plt.figure(figsize=(15,5))

plt.scatter(support, confidence, alpha=0.5, marker="\*")

plt.title('Association Rules')

plt.xlabel('support')

plt.ylabel('confidence')

plt.savefig('static/new\_plot4.png')

plt.clf()

plt.cla()

plt.close()

######################

#Graph Generation - Association Rules

rules\_to\_show =20

import networkx as nx

plt.plot()

G1 = nx.DiGraph()

color\_map=[]

N = 50

colors = np.random.rand(N)

strs=['R0', 'R1', 'R2', 'R3', 'R4', 'R5', 'R6', 'R7', 'R8', 'R9', 'R10', 'R11']

for i in range (rules\_to\_show):

G1.add\_nodes\_from(["R"+str(i)])

for a in rules.iloc[i]['antecedents']:

G1.add\_nodes\_from([a])

G1.add\_edge(a, "R"+str(i), color=colors[i] , weight = 2)

for c in rules.iloc[i]['consequents']:

G1.add\_nodes\_from([c])

G1.add\_edge("R"+str(i), c, color=colors[i], weight=2)

for node in G1:

found\_a\_string = False

for item in strs:

if node==item:

found\_a\_string = True

if found\_a\_string:

color\_map.append('yellow')

else:

color\_map.append('green')

edges = G1.edges()

colors = [G1[u][v]['color'] for u,v in edges]

weights = [G1[u][v]['weight'] for u,v in edges]

pos = nx.spring\_layout(G1, k=16, scale=1)

nx.draw(G1, pos, edges=edges, node\_color = color\_map, edge\_color=colors, width=weights, font\_size=16, with\_labels=False)

for p in pos: # raise text positions

pos[p][1] += 0.07

nx.draw\_networkx\_labels(G1, pos)

# plt.show()

plt.savefig('static/new\_plot5.png')

plt.clf()

plt.cla()

plt.close()

#Delay for saving generated outputs so as could be fetched by HTML.

import time

time.sleep(5) # Delays for 5 seconds. You can also use a float value.

return render\_template('out.html', name = 'Top 5 most selling items', url ='new\_plot1.png', name1 ='abc' , url2='new\_plot2.png', url3='new\_plot3.png', url4='new\_plot4.png', url5='new\_plot5.png')

1. **HTML code for output**

<!doctype html>

<html>

<body style="text-align: center;">

<h1>Dashboard</h1>

<img src= "{{ url\_for('static', filename=url) }}" alt="Chart" >

<br>

<p>{{ name2 }}</p>

<img src= "{{ url\_for('static', filename=url2) }}" alt="Chart" >

<br>

<p>{{ name2 }}</p>

<img src= "{{ url\_for('static', filename=url3) }}" alt="Chart" >

<br>

<p>{{ name2 }}</p>

<h4 style="text-align: center;">Final Output</h4>

<img src= "{{ url\_for('static', filename=url4) }}" alt="Chart" >

<br>

<img src= "{{ url\_for('static', filename=url5) }}" alt="Chart" >

</body>

</html>

1. **Steps to run**

Create two directories - static & template.

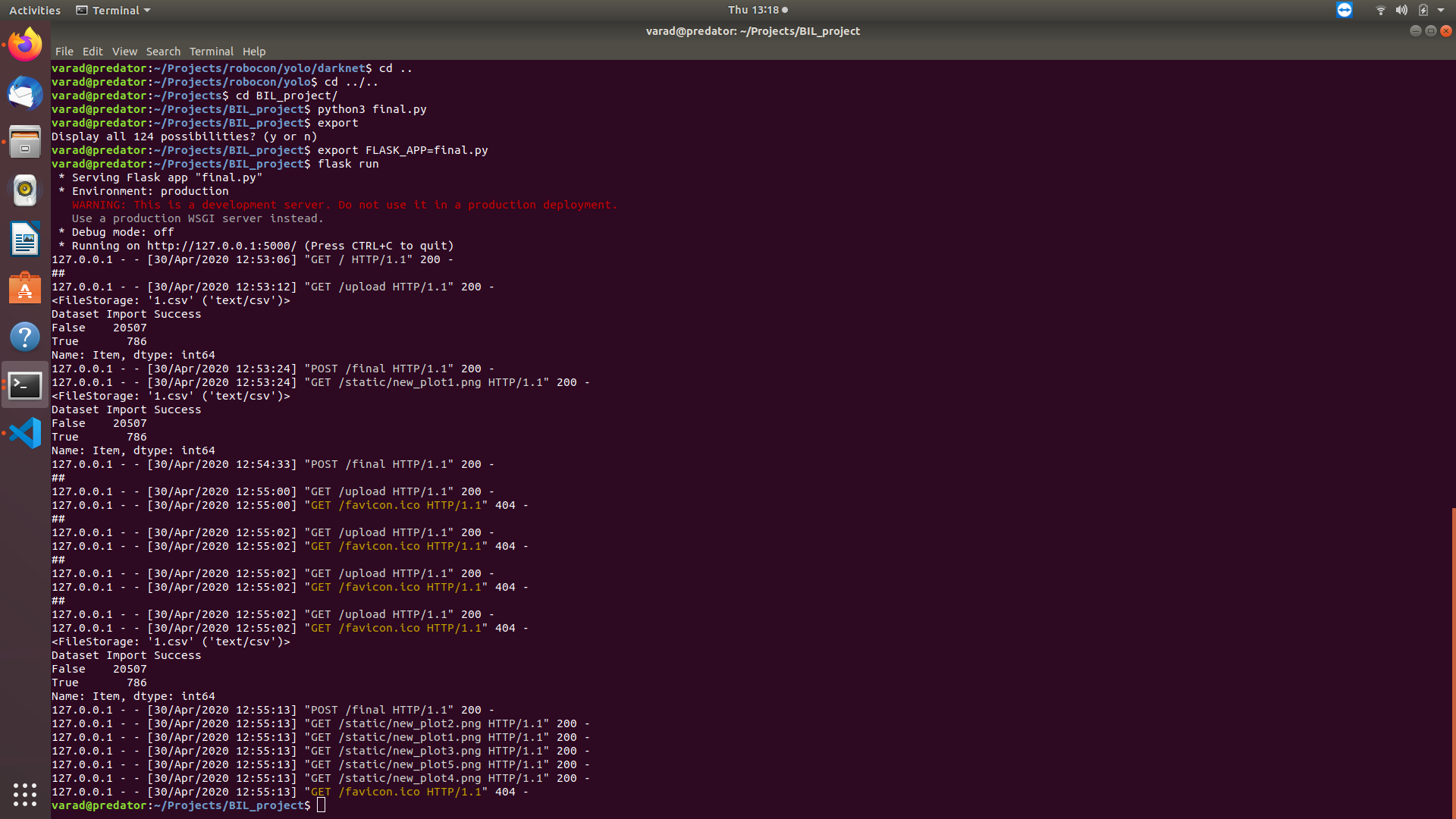
$ mkdir static template

Paste html code as ‘out.html’ in template. And python code in root directory

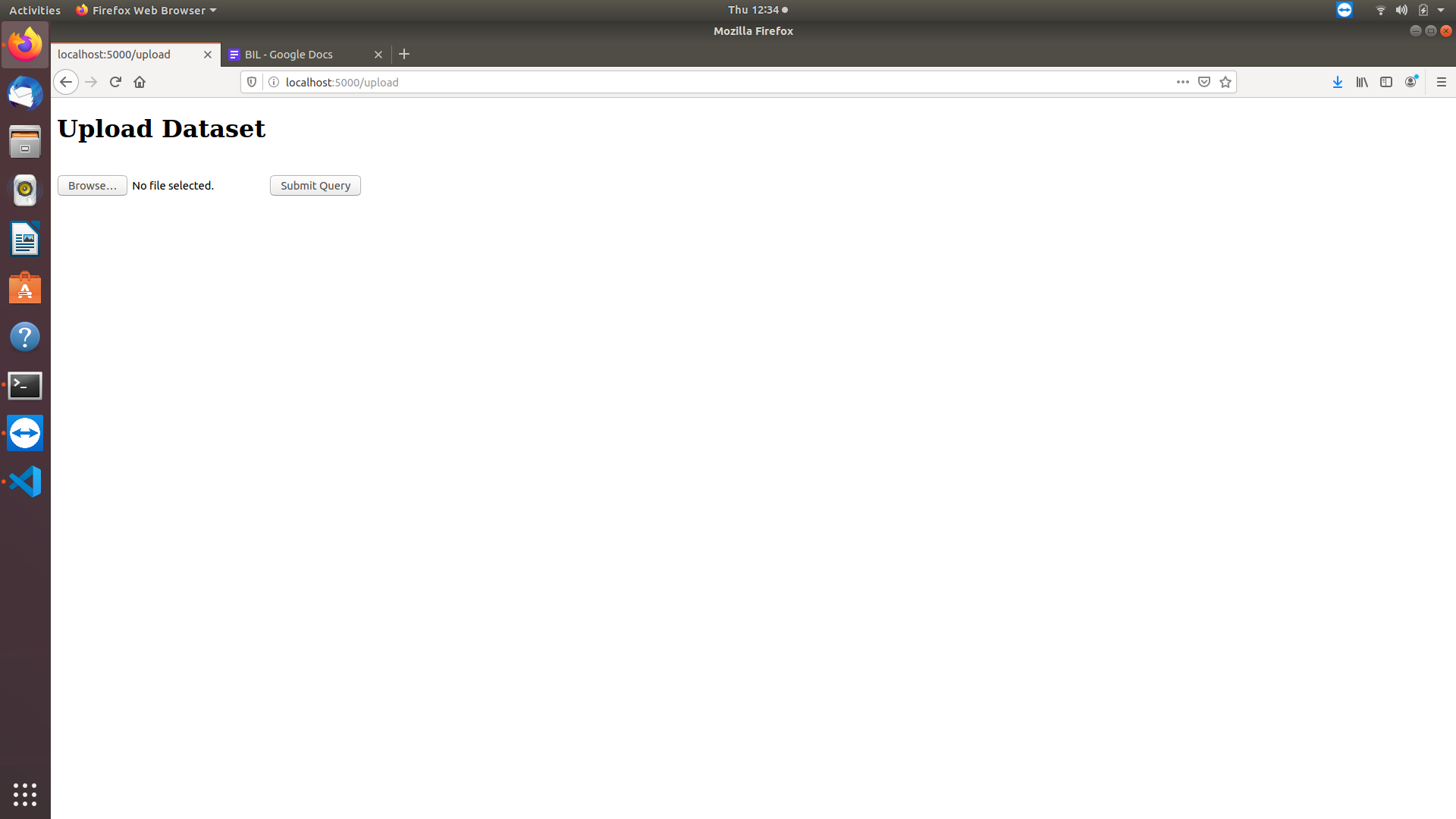
**For Running**

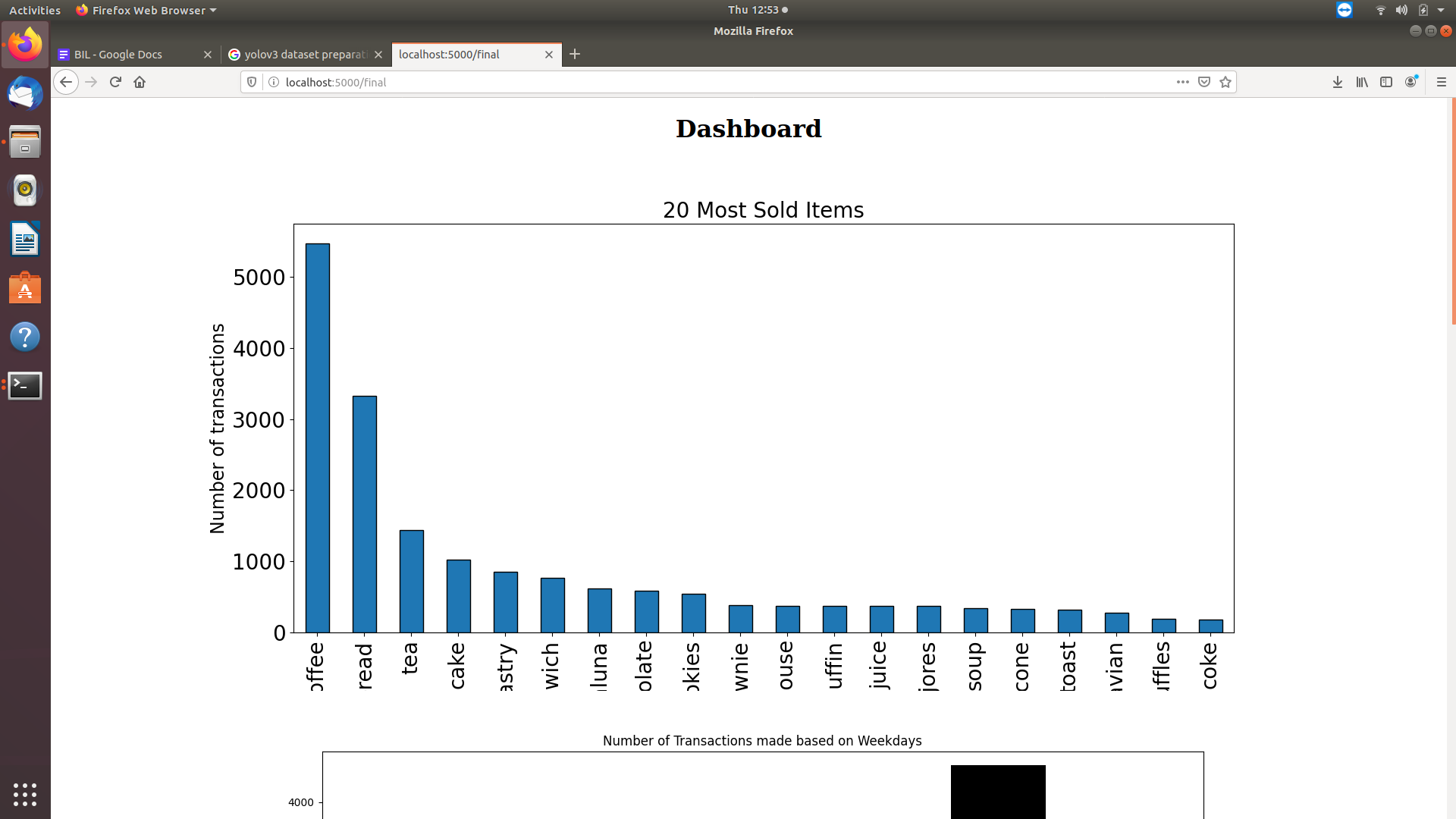
$ export FLASK\_APP=final.py

$ flask run

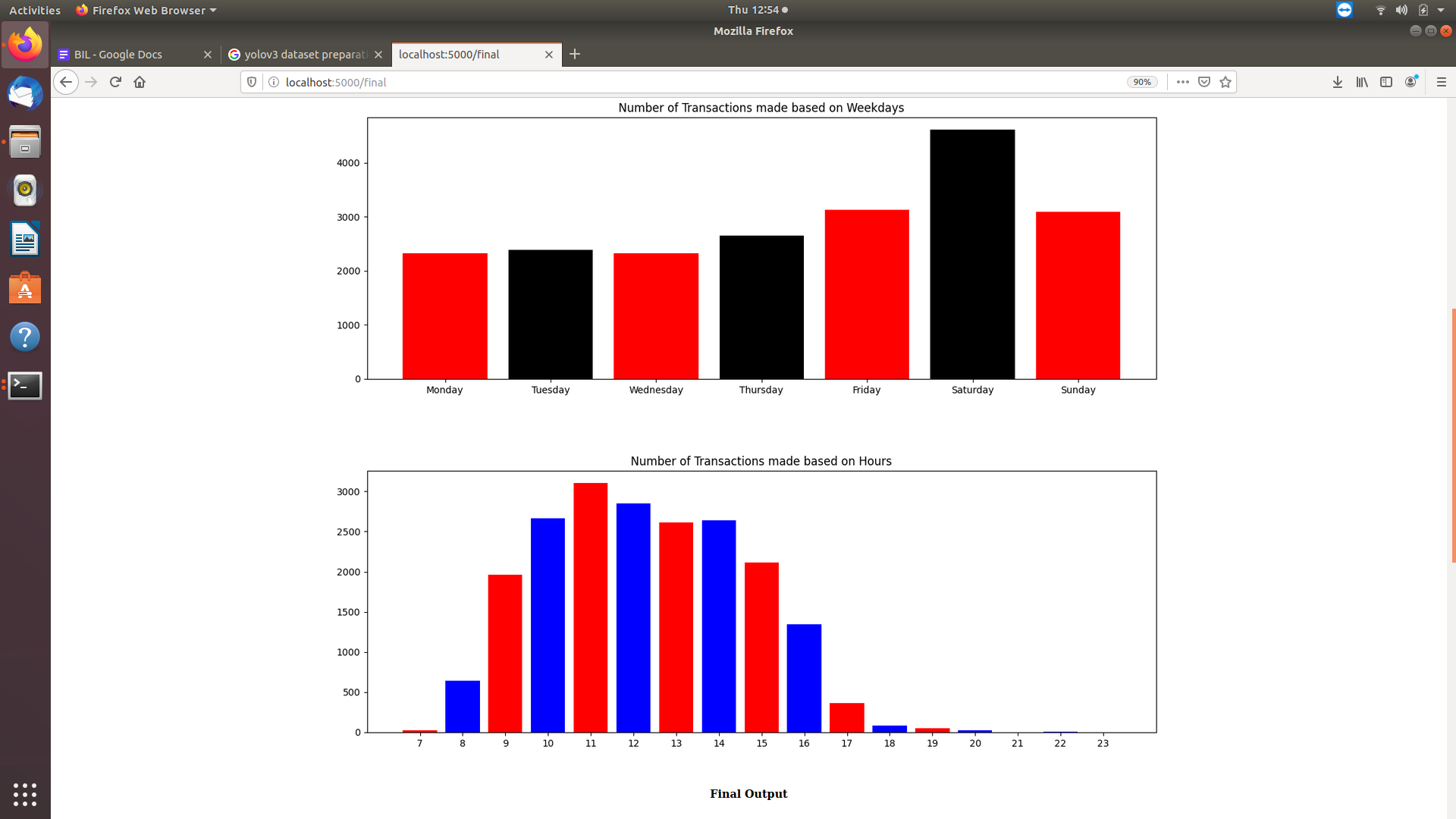
**Expected output on execution**

**Output with Visualizations:**

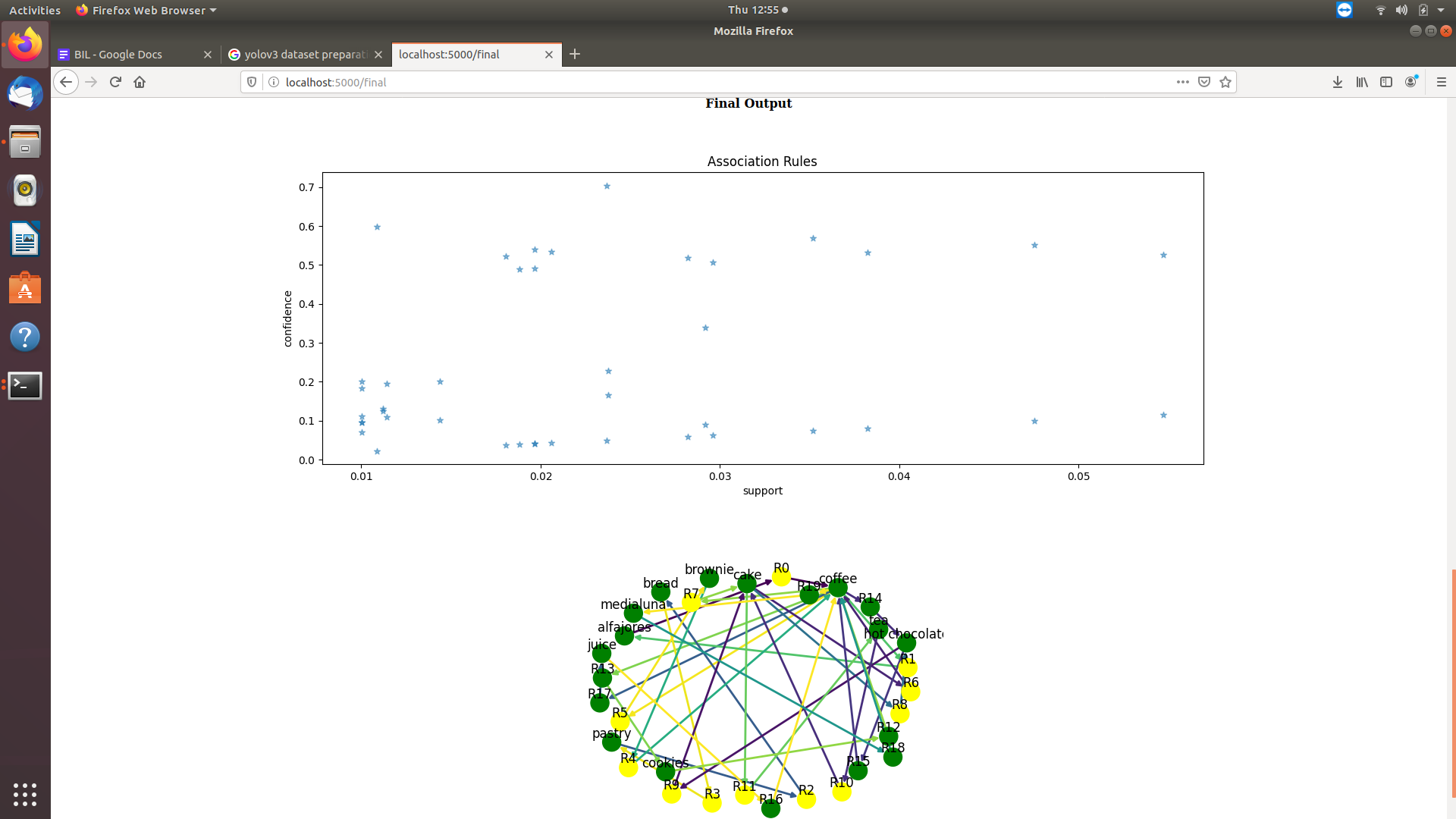
1. **Uploading Dataset**
2. **Dashboard of visualized Results**

****

Top 20 Most sold items which would be required to maintain stock.

****

Based on above visualizations it can be concluded that people make more transactions towards weekends and during peak hours(9am-3pm).



The above graph depicts association rules and apriori algorithm is used to get insight that if a customer buys one item which item he/she buys next.

The association rules for the above dataset can be described as follows:

* ‘cake’ -> R11 -> ‘ hot chocolate’. Hence cake and hot-chocolate are likely to be brought together.
* ‘brownie’-> R4 -> ‘coffee’. Hence brownie and coffee are likely to be brought together.
* ‘coffee’ -> R13 ->’ cookies’. Coffee and cookies are likely to be brought together.

**Conclusion/Analysis of your Project:**

Thus our system is able to perform association mining for predicting the customer’s next purchase. And help the store owner to maintain required stock on the basis of user activities. Moreover, the supervisor can prepare stock refills according to crowd timings and prepare in advance.

**References:**

<https://www.softwaretestinghelp.com/apriori-algorithm/>

<https://www.kaggle.com/ammarikhlaq/eda-and-apriori-alogrithm/data>

<https://www.kaggle.com/aboliveira/bakery-market-basket-analysis>

<https://www.geeksforgeeks.org/apriori-algorithm/>

<https://dwgeek.com/mining-frequent-itemsets-apriori-algorithm.html/>