Project report on

E-Commerce Supply Chain: Blockchain, Smart Contracts & A Packaging Perspective.

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

In

Mechanical Engineering

Submitted by:

Abilash Reddy Gaddam (193226)

Akshay Nuthanapati (193104)

Sushyanth Sridhar (193176)

Vishnu Vardhan (193110)

Under the esteemed guidance of

Prof. Lanka Krishnanand

Professor, Mechanical Engineering

Department of Mechanical Engineering

DEPARTMENT OF MECHANICAL ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY

(An Institution of National Importance)

DEPARTMENT OF MECHANICAL ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY

(An Institution of National Importance)

WARANGAL, TELANGANA STATE-506004

CERTIFICATE

This is to certify that this is the Bonafide record of the project work on "*E-Commerce Supply Chain: Blockchain, Smart Contracts and a Packaging Perspective*" carried out by **Abilash Reddy Gaddam(193226), Akshay Nuthanapati(193104), Sushyanth Sridhar (193176) and Vishnu Vardhan (193110)** of Final year B.Tech (Mechanical Engineering) during the academic year 2022-2023 in the partial fulfillment of the requirements for the award of the Degree of the Bachelor of Technology.

Project Guide

Dr. Lanka Krishnanand

Professor (HAG)

Department of Mechanical Engineering

National Institute of Technology

Warangal – 506004

Head of the Department

Prof. Suresh Babu V

Professor

Department of Mechanical Engineering

National Institute of Technology

Warangal - 506004

APPROVAL SHEET

This Project Work entitled "*E-Commerce Supply Chain: Blockchain, Smart Contracts and a Packaging Perspective"* carried out by by **Abilash Reddy Gaddam(193226), Akshay Nuthanapati(193104), Sushyanth Sridhar (193176) and Vishnu Vardhan (193110)** is approved for the Degree of Bachelor of Technology, Mechanical Engineering.

_____

Dr. Satyanand Abraham,
Faculty Co-Ordinator,
Assistant Professor, MED

_____

Prof. N. Selvaraj
Project Committee Chairman,
Professor (HAG), MED

_____

Supervisor:

_____

Dr. Lanka Krishnanand

Professor (HAG) MED, NITW

Head of Department:

_____

Prof. Suresh Babu V.

Professor, MED, NITW

Date:_____ Place:

_____

DECLARATION

We declare that this written submission represents our ideas in our own words and where other ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will cause disciplinary action by the Institute and can also invoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Abilash Reddy Gaddam)
Roll No:193226

(Akshay Nuthanapati)
Roll No:193104

(Sushyanth Sridhar)
Roll No:193176

(Vishnu Vardan)
Roll No:193110

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to **Dr. Lanka Krishnanand, Professor,**Department of Mechanical Engineering, National Institute of Technology. Warangal. We have been fortunate to undertake this project under her expertise. We always extend our wholehearted gratitude to her for immense support during the project. We also extend our gratitude to **Suresh Babu V. Professor and Head**, Department of Mechanical Engineering, National Institute of Technology, Warangal, for providing us with this opportunity. Finally, we would like to thank **Dr. P R C Gopal**, Assistant Professor, School of Management, National Institute of Technology for guiding us all throughout in our attempt for successful completion of this project title **"E-Commerce Supply Chain: Blockchain, Smart Contracts & A Packaging Perspective".**

Abilash Reddy Gaddam(193226)

Akshay Nuthanpati (193104)

Sushyanth Sridhar (193176)

Vishnu vardhan (193110)

Abstract

Keywords: Packaging, e-commerce, supply chain management, cost-effectiveness.

With the advent of the internet and rapid digitization in the late 20th century, E-Commerce has explodedin popularity and has gradually replaced the conventional supply chain model. Thousands of brick-and- mortar stores across the world have permanently shut down unable to keep up with the cost effectiveness and transportation efficiency of e-commerce supply chains. Almost every retail business today considerse-commerce to be an inescapable part of its business stratagem and thus it is essential for every aspect ofecommerce to be as efficient as possible. E-commerce presents us with a unique opportunity to make theworld a better place by reducing time and distance barriers. However, it also presents challenges in supply chain management with regard to packaging in particular. Increased usage of web-based commerce puts additional pressure on packaging requirements to guarantee safety, storage efficiency and transportation efficacy. This increased onus on packaging could have potentially devastating consequences with regard to the environment and sustainability goals. We have modelled the impact each packaging variable has on the total packaging cost through the use of machine learning techniques.

With the advent of new technologies like Sentiment Analysis and Blockchain various problems faced by stakeholders in the supply chain can be solved. Sentiment Analysis helps manufacturers and distributors understand various problems faced by their customers and the necessary steps to be taken to address these problems. Two methods of Sentiment Analysis are implemented: 1.) A Rules-Based Approach base on probability and an Automatic Approach based on Machine Learning Algorithms. Blockchain provides an easy method to keep a record of the details and help track various goods as they proceed through the Supply Chain. This was accomplished through the use of Smart Contracts – self-executing programs that run when predetermined conditions are met. We have used the Solidity programming language to implement the specific smart contracts for each use case.

It is thus of paramount importance to understand the importance of packaging, Sentiment Analysis and Blockchain in an e-commerce supply chain. We attempt to explain the various topics described below.

CONTENTS

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

One of the most significant developments brought about by the existence of the Internet is e-commerce. It removes constraints related to location or time for the exchange of goods. E-commerce opens up new possibilities, but it also introduces new difficulties in managing the supply chain, particularly with regard to the packaging system. The increasing needs imposed by e-commerce (such as increased quantities of packaging materials for each product, a greater need to preserve products, end-life management, environmental sustainability, etc.) have driven packaging functions to evolve.

Companies have recently started to view packaging as a crucial problem for increasing their global performance and lowering their costs.
One of the main justifications for spending more money on packaging is to lower the likelihood of damage, spoilage, or loss due to theft or lost goods. Regarding logistics, a number of difficulties have raised the profile of packaging. For instance, the relevance of packing has expanded with the rising use of information technology and automation in warehousing and material handling. The management of the entire supply chain may benefit from good packaging. The second use of packaging is mostly a marketing purpose that has to do with brand communication, customer service, and sales promotion. Additional requirements for packaging, particularly in connection to the environment, were imposed in the 1990s. Customers are calling for packaging that is more environmentally friendly and less wasteful, such as packaging that is recyclable or reusable. With the advent of web operations, packaging's role and duties have evolved. Electronic commerce, sometimes known as e-commerce, is a promising application of information technology that has advanced significantly in recent years. It is revolutionising supply chain management, as well as the packaging system, moving away from the old "shop window" approach to become a new form of product protection and containment. It has immense promise for manufacturing, retail, and service operations. Web

operations are among all supply chain operations to play a significant part in developments in the global purchasing process. In recent years, an increasing number of people have started using the Internet and making a variety of purchases there.

Traditional businesses' operations and commercial strategies have evolved as a result of e-commerce channels. Three major problems—integration, customization, and internationalization—have drawn attention to this impact. E-commerce networks, in the first place, enhance value chain integration by lowering transaction costs, enabling "just in time" (JIT) delivery, and enhancing data gathering and processing. Second, extensive levels of product and service customization are made possible through e-commerce databases and direct connections between suppliers and consumers. Finally, the global reach of the Internet enables small businesses to reach clients anywhere.

The main packaging requirements that a company should consider before starting an e-commerce business are:

• **Protection**: Packaged goods must be safeguarded against mechanical shock, vibration, electrostatic discharge, compression, etc. The usage of accessories (such as bubble wrap, air cushions, polystyrene chips, etc.) is often how this is accomplished. Korzeniowski claims that the main purpose of packing in e-commerce is to shield products from physical, chemical, and biological harm. By choosing the right packaging materials, packaging designs, and packaging accessories, adequate protection of goods can be achieved. Packages must have closures that are sufficiently tight, long-lasting, and simple to open and close in order to be properly protected. This is crucial in situations where products do not match customer expectations and must be repackaged and returned.

• **Handleability**: It is necessary to take into account the ergonomic component, which includes all adjustments to the human body and behaviour when using the product. According to a research by Regattieri et al. on how end users perceive packaging, handleability—including ease of handling, simplicity of opening, and userfriendliness—is the primary condition a package should meet.

• **Security**: Packages must ensure secure shipping. It could be necessary to install identification technologies such as RFID tags or barcodes in packages in order to reduce theft, increase security and minimize time spent on the traceability of products.

• **Respect for the environment**: E-commerce produces more waste materials than traditional commerce, because of more frequent orders in smaller quantities. In order to have a minimal environmental impact, it may be necessary for companies to try to recycle packages and minimize dangerous substances emitted when packaging waste is disposed of.

• **Re-use**: Ever more companies have begun re-using packages to ship products to end consumers to minimize both environmental impact and costs. The re-use of packages could also increase customer satisfaction as a result of the low level of environmental pollution produced.

1.2 Aim and Objectives

- After intense literature review, we have identified several variables which contribute to the total cost of packaging. Our aim is to analyze the impact each of these variables has on the total cost of packaging from the standpoint of a particular industry. In this case we have chosen the electronics industry to generate preliminary data.
- Performing sentimental analysis on amazon product reviews and finding out the overall sentiment whether it is positive or negative.
- Finally we will be implementing smart contracts as apart of supply chain in terms of achieving objectives like :

  1.Determining the Provenance(source) of goods.

  2.Tracking the progress of goods through the supply chain.

  3.Building trust through an *open database* of supply chain partners, including their reputation.

Chapter 2

Literature Review

From four different perspectives, Kalakota and Whinston defined e-commerce: (a) communication (e-commerce is the delivery of information, products/services, or payments over telephone lines, computer networks, or any other electronic means); (b) business process (e-commerce is the application of technology towards the automation of business transactions and work flows); and (c) service (e-commerce is a tool that addresses the desire of firms, consumers, and other stakeholders).

The development of e-commerce has also had a big impact on the way the packaging system works. In order to increase the market share of ecommerce, Kathman claims that the Internet is a "highly self select environment" with a new packaging design strategy that must be six distinct from conventional thinking. Visser further claimed that it is challenging to adapt the current packaging design used for the conventional means of purchasing in a physical store as well as marketing strategies into online shopping.

2.1 Identification of Variables from Literature Review

After an extensive literature review and a thorough research we have identified the eight most common and frequently used variables which impact the total cost of packaging.

Table 2.1.1 Identification of Variables from Literature Review

Variables	Meaning/Defintion	References
Material Cost	Cost of the Packaging Material (Cost of corrugated material etc)	Regattieri Alberto et al (2014)
Storage & Handling Cost	Cost of handling & storing bulky packages, heavy materials of construction, drums etc in warehouses.	Zhang Xiaodong et al (2019)
Packaging Operations Cost	This includes all the labour cost like cleaning the package, product filling, closing, labelling, stenciling etc.	Regattieri Alberto et al (2014)

Accessory Cost	This includes all the accessories which are included in packaging for safety, aesthetic purposes etc. Ex:- Air Pillows, Packing Peanuts, Styrofoam etc.	Regattieri Alberto et al (2014)
Transport Cost	Proportion of transport cost which is due to packaging.	Zhang Xiaodong et al (2019)
Cost of Return	Cost incurred when a product is returned.	Zhang Xiaodong et al (2019)
Insurance Cost	It varies depending on the vulnerability of the package to cover the risk of transportation.	Regattieri Alberto et al(2014)
Obsolescence Cost	Cost involved during changes in packaging materials, packages & labels.	Regattieri Alberto et al (2014)

Chapter 3

Impact of packaging variables on the total cost of packaging

3.1 Methodology

The way we aim to predict the impact each of the variables has on the total cost of packaging is by applying Machine Learning models. There are various regression models like Multiple Linear Regression, Polynomial Linear Regression, Decision Tree Regression & Random Forest Regression.

The literature review we performed showed us just how important packaging is for an e-commerce supply chain. Packaging is one of the few areas where e-commerce Literature Review Identification of Variables Model Building for Total Cost Preliminary Data Collection Real Time Data Analysis 11 supply chains still lag a traditional one. It is thus crucial for companies to estimate the total contribution of packaging to e-commerce supply chains. From our literature review we have identified 6 of the most frequently occurring variables. These variables were deemed to be crucial for predicting the total cost of packaging. We will then attempt to build a model to predict total cost of packaging by determining the contribution of each of these 6 individual variables.

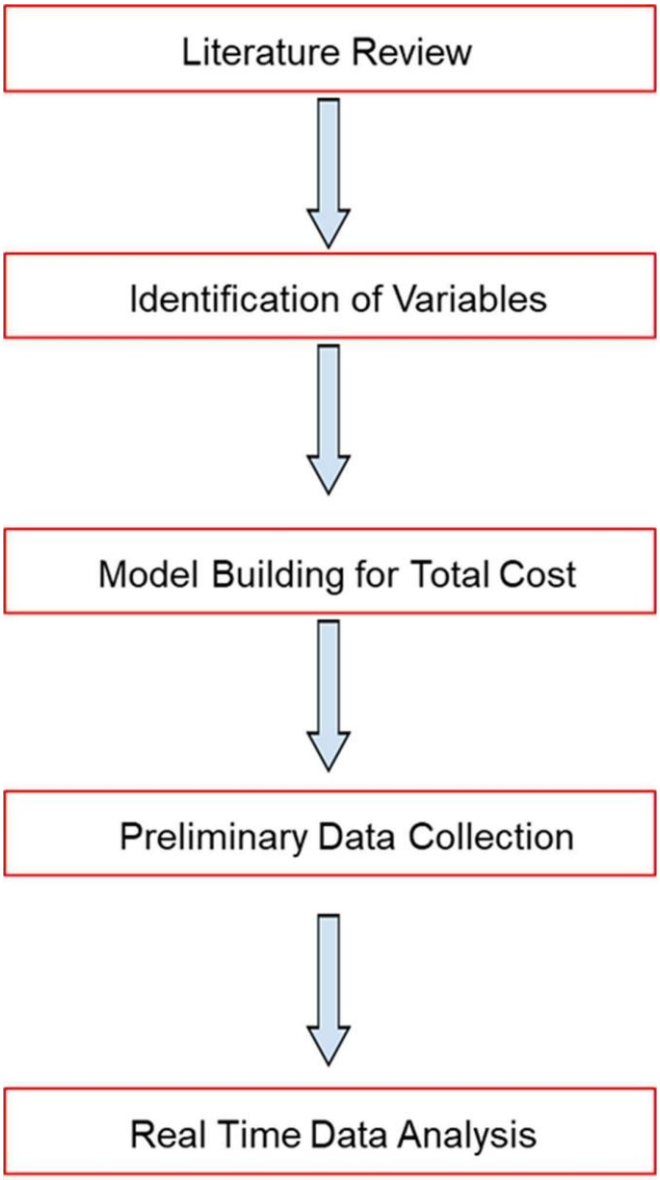

Fig. 3.1.1 Flowchart of Machine Learning Model Implementation of Packaging Variables.

If we were to build a multiple linear regression model, our equation would look similar to this:

$$y = b_0 + b_1*x_1 + b_2*x_2 + \ldots + b_n*x_n$$

The different betas would give us the weights of each individual variable (contribution of that variable to the total packaging cost). Multiple linear regression works on the principle of fitting a surface to minimize the discrepancy (error) between predicted & actual output values.

Λ similar equation for polynomial regression would be:

Y= b0+b1x1+ b2x1 2+ b2x1 3+..... bnx1 n

Our next step involves data collection. For this step we were able to collect preliminary data from external expert sources. We obtained an approximate range of values for each variable. The final ranges for each variable utilized in our project are as follows: Material cost: ₹25-270 Storage & Handling Costs: ₹50-60 Packaging Operations Costs (Labor Cost): ₹6-21 Cost of Packaging Accessories (Accessory Cost): ₹8-90 Transportation Cost of Filled Packages: ₹16-180 Cost of return/repackaging: ₹0-100.

Using the above ranges as upper and lower limit we have generated a hundred random variables between these two limits.

3.2 Code

The machine learning library scikit learn was used to perform linear, polynomial, decision-tree and random forest regression. Scikit-learn is a powerful open-source machine learning software which allows us to split our data for training and testing. We have split our data set in a 75-25 manner with 75% of our data being used for training the algorithm and the remaining for testing its performance.

Linear Regression

```
In [251]:
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(df,df1,test_size=0.25)

In [252]:
from sklearn.linear_model import LinearRegression

In [253]:
linreg = LinearRegression( )

In [254]:
linreg.fit(x_train,y_train)

Out[254]:
LinearRegression()

In [255]:
y_pred_1 = linreg.predict(x_test)

In [256]:
linreg.coef_

Out[256]:
array([1.00854535, 0.3766424 , 0.84144387, 1.01432797, 1.04990792,
       1.06615147])

In [257]:
linreg.intercept_

Out[257]:
103.0047207294715

In [258]:
import numpy as np
from sklearn import metrics
print(100 - np.sqrt(metrics.mean_squared_error(y_pred_1,y_test)))

83.51876967944209
```

Fig. 3.2.1 Code for Linear Regression Model

Decision Tree regressor

```
In [263]:
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor()

In [264]:
regressor.fit(x_train,y_train)

Out[264]:
DecisionTreeRegressor()

In [265]:
y_pred_2 = regressor.predict(x_test)

In [266]:
y_pred_2

Out[266]:
array([447.9149501, 492.8791953, 430.6874234, 360.9350362, 354.8926672,
       433.8097157, 509.6889   , 484.2286878, 408.9324832, 396.2335908,
       509.6889   , 578.8746757, 484.2286878, 492.8791953, 389.4135593,
       447.9149501, 396.2335908, 564.965496 , 545.6367167, 583.4372863,
       440.4339028, 366.0160746, 424.3877383, 366.0160746, 360.9350362])

In [267]:
from sklearn import metrics
print(100 - np.sqrt(metrics.mean_squared_error(y_pred_2,y_test)))

53.56622623090173
```

Fig. 3.2.2 Code for Decision Tree Regressor Model

Polynomial & Random Forest Regressor

```
In [268]:
1 x = df
2 y = df1

In [269]:
1 from sklearn.preprocessing import PolynomialFeatures
2 from sklearn.linear_model import LinearRegression
3 poly_reg = PolynomialFeatures(degree = 2)
4 X_poly = poly_reg.fit_transform(x)
5 lin_reg = LinearRegression()
6 lin_reg.fit(X_poly, y)
7 y_pred_3 = lin_reg.predict(X_poly)

In [270]:
1 print(100 - np.sqrt(metrics.mean_squared_error(y_pred_3,y)))

80.78031030080646

In [271]:
1 from sklearn.ensemble import RandomForestRegressor

In [272]:
1 rf = RandomForestRegressor(n_estimators = 300, max_features = 'sqrt', max_depth = 5, random_state = 18).fit(x_train, y_train

In [273]:
1 y_pred_4 = rf.predict(x_test)

In [274]:
1 print(100 - np.sqrt(metrics.mean_squared_error(y_pred_4,y_test)))

64.10514398379635
```

Fig. 3.2.3 Code for Polynomial and Random Forest Regressor Model

Chapter 4

Sentimental analysis of amazon product reviews

4.1 Methodology

The methodology of this study entails primarily two steps: the inputting of the 'pack list' and the classifying of provided comments by users (customers) into two categories based on their nature. The pack list is the set of words that is fed to the algorithm as a means to enable the system to perform unsupervised learning on the comments. The two categories by which the comments will be distinguished are positive and negative; the latter being a subject of worry and an area of scope of improvement for the currently employed design of the packaging.

The methodology is explained below using a flowchart:

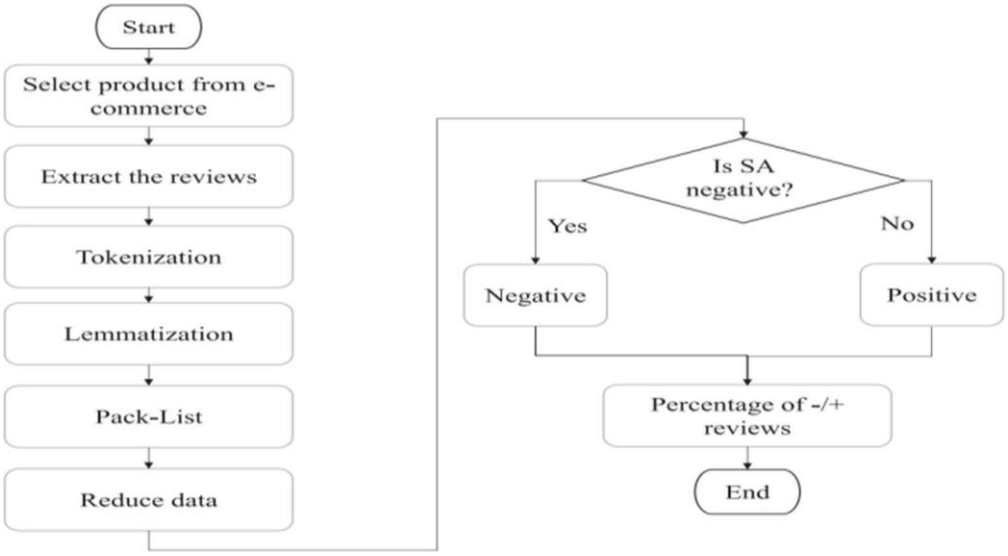

Fig. 4.1.1 Flowchart for Implementation of Sentiment Analysis

A list of package-related words, called Pack-List, was needed to identify the packaging-related feedback. To create a Pack List, the first step was choosing the lowest rating reviews (1- and

2-star ratings) for the targeted products, which can be multiple products in the same category. Next, to find the frequency of each word, these reviews were put in a word count platform. Then, these words have been ranked based on frequency. The words related to packaging and the possible cause of package damage were selected depending on the products The reviews related to the packaging were collected by using Pack-List as the guide. Then, the sentiment of the reviews was realized by using SA.

• The selected product for our study is the electronics industry.

• The selected industry's comments and reviews are extracted using the URL of the outlet's website and code was written in order to web scrap the reviews and finally stored the reviews in CSV file format.

• Tokenization and lemmatization processes were performed which break down a comment into words, characters, and subwords and convert them into their base form.

• The sentiment analysis is performed on the dataset based on which the result is checked whether it is negative or not.

• Calculate the percentages of negative and positive reviews

$$\text{Percentage of Negative or Positive} = \left(\frac{\text{Number of Negative or Positive reviews}}{\text{Total Number of reviews}}\right) \times 100$$

Web scrapping was basically done in order to extract the reviews from the amazon website through its URL. Code was written in order to segregate the entire review data and to transfer it to a CSV file.

A pack list was generated by collecting the lowest rating reviews of targeted products and the words which have high frequency were selected. In this process the following pack list was obtained:

Break, damage, crack, Deliver, Delivery, Destroy, Fail, Failure, Defect, Defective, Distort, Scratch, Protective, Shatter, Screen, Box.

There are two methods by which sentiment analysis is done: 1.) Rule-based approach 2.) Automatic Approach.

Rule-based Approach:

$$
\begin{aligned}
&P(\text{“The package was received broken”}) \\
&= P(The)P(package)P(was)P(received)P(broken)
\end{aligned}
$$

$$
\begin{aligned}
&P(Positive|The\ package\ was\ received\ broken) \\
&= P(The|+) \times P(package|+) \times P(was|+) \times \\
&\quad P(received|+) \times P(broken|+)
\end{aligned}
$$

$$
\begin{aligned}
&P(Negative|The\ package\ was\ received\ broken) \\
&= P(The|-) \times P(package|-) \times P(was|-) \times \\
&\quad P(received|-) \times P(broken|-)
\end{aligned}
$$

We have both a positive and negative training data set.

For instance, if the frequency of the word ‘broken’ in negative training data is 15 and the total number of words in training negative review is 100 the probability of P(broken|-) is 15/100.

The probability of the sentence being positive or negative will be calculated by the two equations respectively. The equation with the higher value shows the sentiment of the sentence.

Automatic Approach:

This approach depends on Machine learning techniques. Here the sentiment analysis task is usually modelled as a classification problem, whereby a classifier is fed a text and returns a category, e.g. positive, negative.

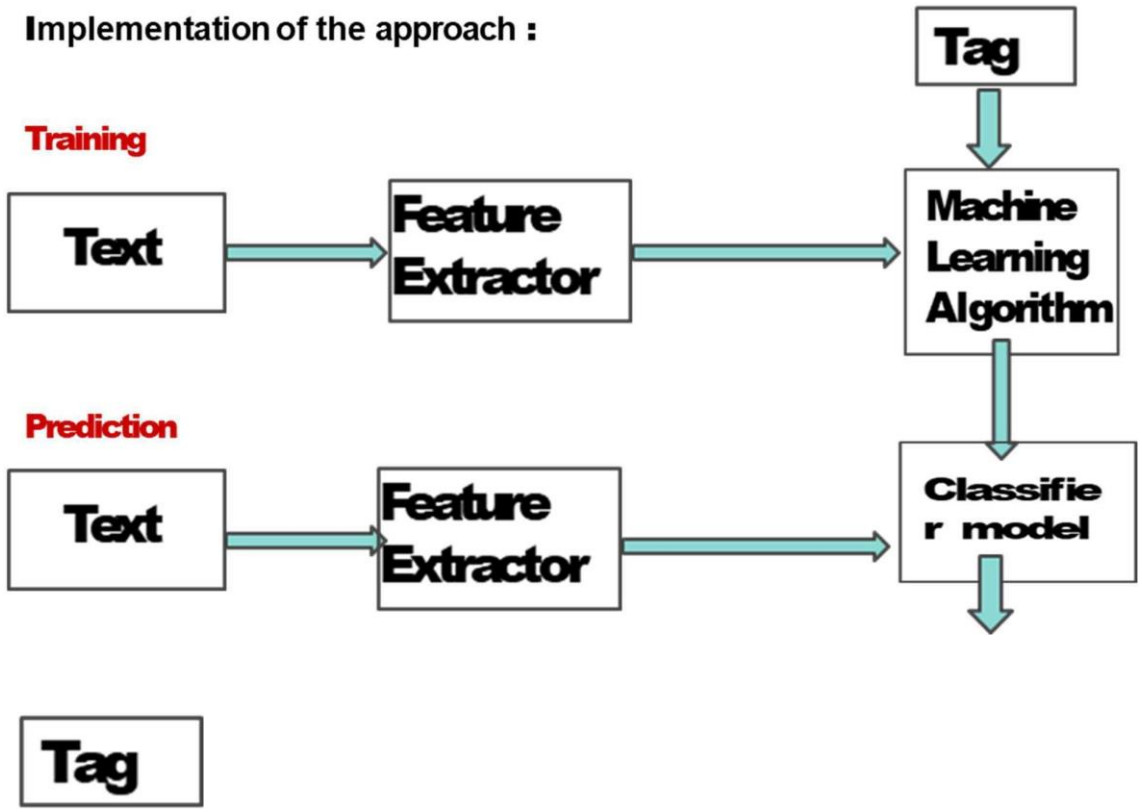

Fig 4.1.2 Implementation of Automatic Approach

4.2 Code

Rules based approach:

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

```
[1] from google.colab import files
    uploaded = files.upload()

    Choose Files  amazon2.csv
    • amazon2.csv(text/csv) - 823006 bytes, last modified: 3/5/2023 - 100% done
    Saving amazon2.csv to amazon2.csv

[2] import pandas as pd
    import io
    #p_test = pd.read_csv('TrainSA.csv')

    df2 = pd.read_csv(io.BytesIO(uploaded['amazon2.csv']))
    df2 = df2.dropna()

[3] temp=df2['Reviews']

[4] col_list = temp.values.tolist()

[5] len(col_list)

    3694

[6] lst = [i.lower() for a,i in enumerate(col_list)]
```

```
x=len(temp)
P1=1
P2=1
tp1=[]
tp2=[]
inp="the screen was damaged"
words = inp.split(' ')
for i in range(0,len(words)):
  y=temp.count(words[i])
  tp1.append(y/x)
for i in range(0,len(words)):
  y=temp1.count(words[i])
  tp2.append(y/x)


for i in range(0,len(tp1)):
  P1=P1*tp1[i]
for i in range(0,len(tp2)):
  P2=P2*tp2[i]

print(P1)
print(P2)
```

```
1.1058914381746564e-10
4.39581097335599e-07
```

Fig. 4.2.1 Code for Rules-Based Approach

Automatic approach:

```
import nltk.classify.util
from nltk.classify import NaiveBayesClassifier
import numpy as np
import re
import string
import nltk

cleanup_re = re.compile('[^a-z]+')
def cleanup(sentence):
    sentence = str(sentence)
    sentence = sentence.lower()
    sentence = cleanup_re.sub(' ', sentence).strip()
    return sentence
senti["Summary_Clean"] = senti["reviews.text"].apply(cleanup)
check["Summary_Clean"] = check["reviews.text"].apply(cleanup)
train["words"] = train["Summary_Clean"].str.lower().str.split()
test["words"] = test["Summary_Clean"].str.lower().str.split()
check["words"] = check["Summary_Clean"].str.lower().str.split()

train.index = range(train.shape[0])
test.index = range(test.shape[0])
check.index = range(check.shape[0])
prediction =  {}
```

```
from wordcloud import STOPWORDS
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
stopwords = set(STOPWORDS)
stopwords.remove("not")
count_vect = CountVectorizer(min_df=2 ,stop_words=stopwords , ngram_range=(1,2))
tfidf_transformer = TfidfTransformer()
X_train_counts = count_vect.fit_transform(train["Summary_Clean"])
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
X_new_counts = count_vect.transform(test["Summary_Clean"])
X_test_tfidf = tfidf_transformer.transform(X_new_counts)

from sklearn.naive_bayes import MultinomialNB
model1 = MultinomialNB().fit(X_train_tfidf , train["senti"])
prediction['Multinomial'] = model1.predict_proba(X_test_tfidf)[:,1]
print("Multinomial Accuracy : {}".format(model1.score(X_test_tfidf , test["senti"])))

from sklearn.naive_bayes import BernoulliNB
model2 = BernoulliNB().fit(X_train_tfidf,train["senti"])
prediction['Bernoulli'] = model2.predict_proba(X_test_tfidf)[:,1]
print("Bernoulli Accuracy : {}".format(model2.score(X_test_tfidf , test["senti"])))

from sklearn import linear_model
logreg = linear_model.LogisticRegression(solver='lbfgs' , C=1000)
logistic = logreg.fit(X_train_tfidf, train["senti"])
prediction['LogisticRegression'] = logreg.predict_proba(X_test_tfidf)[:,1]
print("Logistic Regression Accuracy : {}".format(logreg.score(X_test_tfidf , test["senti"])))

```

```
def test_sample(model, sample):
    sample_counts = count_vect.transform([sample])
    sample_tfidf = tfidf_transformer.transform(sample_counts)
    result = model.predict(sample_tfidf)[0]
    prob = model.predict_proba(sample_tfidf)[0]
    print("Sample estimated as %s: negative prob %f, positive prob %f"
          % (result.upper(), prob[0], prob[1]))

test_sample(logreg, "The product was good and safe")
test_sample(logreg, "the whole experience was just
            horrible and product is worst")
test_sample(logreg, "It is defective and damaged")
```

```
Sample estimated as POS: negative prob 0.012834, positive prob 0.987166
Sample estimated as NEG: negative prob 0.990836, positive prob 0.009164
Sample estimated as NEG: negative prob 0.998993, positive prob 0.001007
```

Fig 4.2.2 Code for Automatic Approach

Chapter 5

Deploying Smart contracts as a part of Supply chain using Solidity

5.1 Methodology

The start of the supply chain will need to add the specifics of its raw materials and its own details to the blockchain. The place and time from which the raw material is transported can be permanently stored by integration with a smart sensor. Using a product's serial number or tag, customers and parties to the supply chain can use the Provenance smart contract to verify a product's provenance. Once the contract has been published to the blockchain, the following functions can be called:

A vast 'eer-to-peer network's resources are used by the distributed ledger technology known as blockchain to verify and approve transactions. Blocks are used to store these transactions chronologically, cryptographically connect the blocks together, and store them permanently on the blockchain to produce an immutable chain.

Principles underlying blockchain technology:

1.) Distributed database

2.) Peer-to-peer transmission

3.) Transparency

4.) Irreversibility

5.) Computational logics (on which we can run smart contracts)

Smart Contracts are self-executing computer programs that automatically enforce the terms of an agreement between two or more parties.

Broad Overview of a generic Supply Chain:

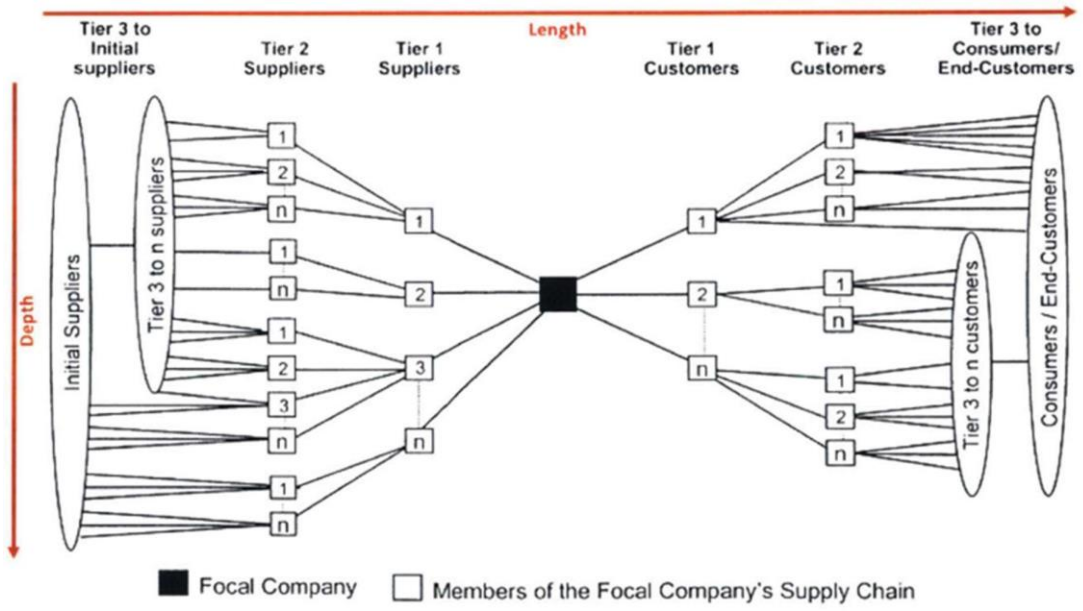

Fig. 5.1.1 Broad Overview of a Generic Supply Chain

Stakeholder Value Network:

Various Stakeholder:

1.) Original Equipment Manufacturer

2.) Raw Material Provider

3.) Component Manufacturer

4.) Inbound Distributor

5.) Contract Manufacturer

6.) Outbound Distributor

7.) Retailer

8.) Consumer

9.) Regulatory Authority

10.) Non-Governmental Organization (NGO)

Fig. 5.1.2 Implementation of Smart Contracts

Objectives:

1. Determining the ***provenance*** (source) of goods.
2. ***Tracking*** the progress of goods through the supply chain.
3. Building trust through an ***open database*** of supply chain partners, including their reputation.

To demonstrate the application of smart contracts, a simplified, generic supply chain for a basic consumer electronic product is used.

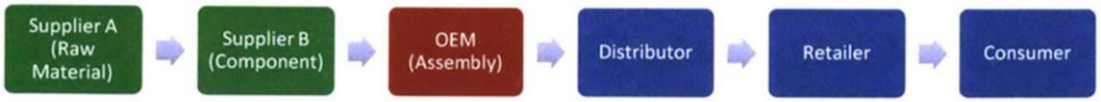

Fig. 5.1.3 Demonstration of Smart Contracts

On Ethereum, an open blockchain platform for creating and utilizing decentralized apps, a proof-of-concept has been created. Provenance, Tracking, and Reputation are the three distinct smart contracts that make up the proof-of-concept. Solidity, an object-oriented programming language that is popular for creating smart contracts, is used to create the contracts.

Establishing the provenance of products.

Supplier A, who started the supply chain, will have to add information about both itself and its raw materials to the blockchain. The place and time from which the raw material is transported can be permanently stored by integration with a smart sensor. Any other parties, including the customer, will be able to access the information as a result via the blockchain. By making it simple to establish provenance, raw material suppliers are held accountable and parties can quickly confirm the product's origin.

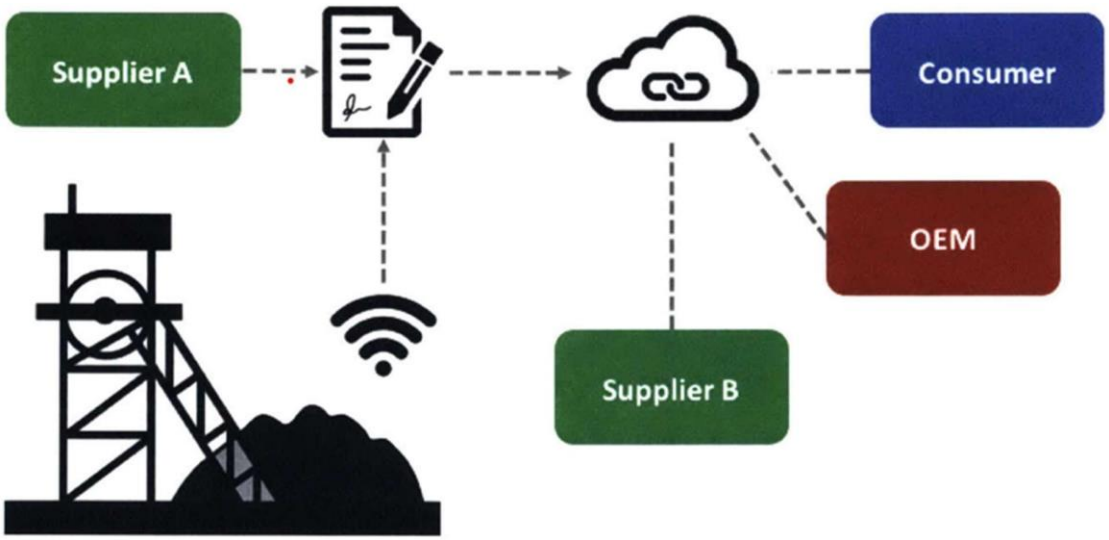

Fig. 5.1.4 Provenance of Products

Using a product's serial number or tag, customers and parties to the supply chain can use the Provenance smart contract to verify a product's provenance. Once the contract has been put into use on the blockchain, the following functions can be used.

1.) addProducer – this function enables suppliers and producers to enter their information and add it to the blockchain. The producer's Ethereum public address is mapped by Solidity to a struct holding information about it, including its name, phone number, city, state, and country of origin as well as certification. Therefore, using the producer's public address, the information can be quickly retrieved afterwards.
2.) removeProducer – This function enables the contract administrator (the focal organisation or a third-party impartial to the contract) to delete a producer or supplier from the database in the event of any alterations. To avoid pointless tinkering, administrator access is necessary.
3. findProducer: By entering a producer or supplier's Ethereum public address, this feature enables anyone to view their details. This can be used by customers to confirm the legitimacy and certification of the producer that is selling their product. Given that the blockchain does not need to be altered, this function is free.

4.) certifyProducer – This lets the contract administrator certified a specific producer or supplier (perhaps after receiving the required paperwork). The blockchain will be used to store and display the certification status along with other producer information. In order to give an additional degree of verification, administrator access is necessary.
5.) addProduct – This feature enables a producer or supplier to log each time their product is tagged at the point of origin on the blockchain. It can track the product's location automatically if a smart sensor is integrated. The block timestamp and producer's public address will be immediately added to the blockchain after the transaction has been added. The serial number or tag of the product is connected to its details using a mapping.
6.) removeProduct – This feature enables the contract administrator to delete a product from the database in the event of any modifications. To avoid pointless tinkering, administrator access is necessary.
7.) findProduct – This tool enables anyone to view a product's data by entering the serial number of the item. Along with the producer's public address, the location, time, and date of origin are also returned. Given that the blockchain does not need to be altered, this function is free. To ascertain the provenance of their products, customers can utilise this service in conjunction with the findProducer function.

Monitoring the movement of goods

If certain prerequisites are satisfied, the Tracking smart contract enables stakeholders in a supply chain to follow the delivery of products and execute payment in the form of tokens after each leg of the shipment is finished. The contract consists of two parts: one is for managing tokens, and the other is for handling shipments.

1) sendToken – By giving the public addresses and the desired token quantity, this function enables the transfer of tokens between Ethereum accounts. It includes functionality built in to determine whether the sender's account has enough tokens and, after the transaction has been successful, to automatically update the balances of both accounts.
2) getBalance – By specifying the Ethereum public address, this function enables any party to view the token balance of an account. Given that the blockchain does not need to be altered, this function is free. The quantity of initial tokens can be determined once the contract is deployed to the blockchain, and all tokens are originally held by the administrator. The determination of the token's worth and the distribution of the initial tokens will require separate agreements.
3) recoverToken – This function enables the contract administrator to retrieve tokens from any account. As the sendToken method can be used to send tokens from any account to another account, it serves as a check and balance to avoid abuse. There is logic built in to verify for a sufficient amount and automatically update balances, similar to the sendToken method.
4) setContractParameters – This function enables the contract administrator to specify the requirements that must be satisfied (and that have been agreed upon by the parties) before a shipment is successful and payment is released. It contains information on the shipping lead time, shipping location, and payment amount (represented as tokens).
5) sendShipment: Once a shipment has been sent, this feature enables the sender to register its specifics on the blockchain. The tracking number or tag of the package is mapped to information like the item and amount using a mapping. Real-time location information can be provided by integrating a smart sensor. The moment of dispatch and the sender's address are likewise recorded to the blockchain. To notify parties that an item has been dispatched, an event is started.

6) receiveShipment: This feature enables the recipient to document a shipment's data on the blockchain once it has arrived. This function employs a two-layer logic structure. The first check is to make sure the product and quantity received match those that were sent. If they do, an event is started to record that the item has been successfully received. The following layer of logic then determines whether the established shipment lead time and destination have been followed. If so, payment to the shipping party will be sent automatically using the sendToken function. If problems occur, such as a payment that was not sent or an item or quantity that did

7) deleteShipment – This function enables the contract administrator to delete a shipment from the database in the event of any modifications. To avoid pointless tinkering, administrator access is necessary.

8) checkShipment: By entering their tracking number, any party can use this tool to view a shipment's data. The item, amount, position (which may be real-time if a smart sensor is integrated), shipment timestamp, and sender are all returned. This function is free because it doesn't involve changing the blockchain in any way; nevertheless, if real-time location data is utilised, it will become expensive.

9) checkSuccess – This function enables any party to check the overall number of successful shipments made by a party in the supply chain. Successful shipments are ones that have adhered to the preset standards outlined in the contract (i.e., weren't late and were delivered to the right place). Given that the blockchain does not need to be altered, this function is free.

10) calculateReputation – This function enables any party to determine a supply chain partner's reputation score. The number of successful shipments divided by the total number of shipments made by a specific party yields the reputation score, which is stated as an integer between 0 and 100. A party's default reputation score is zero if it has not yet made any shipments. Given that the blockchain does not need to be altered, this function is free. Additionally, it is necessary for the Reputation smart contract to operate correctly.

Open Database of Suppliers and Supply Chain Participants

1) addSupplier enables suppliers and other stakeholders to enter their information and add it on the blockchain. The supplier's Ethereum public address is mapped using a mapping to a struct holding information about it, including name, phone number, city, state, and country of origin, sort of commodities it specialises in, and reputation. As a result, using the supplier's public

address, the information can be quickly obtained afterwards. The address is also added to an array that holds the addresses of all the parties.

2) removeSupplier – in the event of any changes, this function enables the contract administrator to delete a party or supplier from the database. Additionally, its public address is taken out of the list of consolidated addresses. To avoid pointless tinkering, administrator access is necessary.

3) findSupplier: By providing their Ethereum public address, any party can use this tool to display a supplier's or party's information. Due to the availability of information regarding the sort of goods and reputation, parties can use this to search acceptable and preferred providers. Given that the blockchain does not need to be altered, this function is free.

4) allSuppliers: This feature enables any party to view the whole list of suppliers and other parties that have been registered on the blockchain. The list of Ethereum public addresses is returned as an array, which can be used by others to obtain more information. Given that the blockchain does not need to be altered, this function is free.

5) filterByGoodsType – This feature enables anyone to look for suppliers based on the categories of goods in which they have expertise. The entire supplier array is converted into a memory array that is the same size. Iterating over the entire supplier array, logic is used to look for suppliers whose items types match the one mentioned in the query. The new array is then filled with the matches 55. Given that the blockchain does not need to be altered, this function is free.

6) filterByReputation – Like the preceding function, this one enables any party to look for providers based on the reputation score they have. The entire supplier array is converted into a memory array that is the same size. In order to find suppliers with reputation scores that are equal to or greater than the score supplied in the query, logic is utilised to iterate across the entire supplier array. The new array is then returned with the matches. Given that the blockchain does not need to be altered, this function is free.

7) checkReputation: By entering a supplier's or party's Ethereum public address, this function enables any party to view the reputation score of that supplier or party. Calling the deployed Tracking contract and executing the calculateReputation function are how it operates. Both operations are free because they don't involve altering the blockchain in any way.

8) updateReputations: This command enables the contract's administrator to calculate the most recent updates to the reputation scores of all parties and to update those scores on the blockchain. The reputation scores may become out of current when more shipments are made because they are saved when parties submit their contact information using the addSupplier

function. Therefore, the administrator can make advantage of this feature to routinely update the reputations. Using the calculateReputation method from the Tracking contract, it iterates over the entire supplier array while updating the reputation score for each public address. In order to avoid unauthorised usage, administrator access is assigned.

5.2 Code

A.) Provenance Smart Contract Code

```
contract Provenance {

    address admin;
    mapping (address => Producer) producers;
    mapping (string => Product) products;

    struct Producer {
        string name;
        uint phoneNo;
        string cityState;
        string country;
        bool certified;
    }

    struct Product {
        address producer;
        uint[] locationData; // array containing lat & long
        uint timeStamp;
    }

    // constructor - runs once when contract is deployed
    function Provenance() {
        admin = msg.sender;
    }

    // modifier to allow only admin to execute a function
    modifier onlyAdmin() {
        if (msg.sender != admin) throw;
        _;
    }
```

```
    // function for producer to add their details to database
    function addProducer(string _name, uint _phoneNo, string
_cityState, string _country) returns (bool success) {

        // don't overwrite existing entries and ensure name isn't null
        if (bytes(producers[msg.sender].name).length == 0 &&
bytes(_name).length != 0) {
            producers[msg.sender].name = _name;
            producers[msg.sender].phoneNo = _phoneNo;
            producers[msg.sender].cityState = _cityState;
            producers[msg.sender].country = _country;
            producers[msg.sender].certified = false;


            return true;
        }
        else {
            return false; // either entry already exists or name
entered was null
        }
    }

    // function to remove producer from database (can only be done by
admin)
    function removeProducer(address _producer) onlyAdmin returns (bool
success) {
        delete producers[_producer];
        return true;
    }

    // function to display details of producer
    function findProducer(address _producer) constant returns (string,
uint, string, string, bool) {
        return (producers[_producer].name,
producers[_producer].phoneNo, producers[_producer].cityState,
producers[_producer].country, producers[_producer].certified);
    }

    // function to certify producer as legitimate (can only be done by
admin)
    function certifyProducer(address _producer) onlyAdmin returns
(bool success) {
        producers[_producer].certified = true;
        return true;
    }
```

```
    // function for producer to add their product to database
    function addProduct(string serialNo, uint[] _locationData) returns
(bool success) {

        // ensure no duplicate serial numbers and serial number isn't
null
        if (products[serialNo].producer == 0x0 &&
bytes(serialNo).length != 0) {
            products[serialNo].producer = msg.sender;
            products[serialNo].locationData = _locationData;
            products[serialNo].timeStamp = block.timestamp;
            return true;
        }
        else {
            return false; // either serial number already in use or
serial number entered was null


        }
    }

    // function to remove product from database (can only be done by
admin)
    function removeProduct(string serialNo) onlyAdmin returns (bool
success) {
        delete products[serialNo];
        return true;
    }

    // function to display details of product
    function findProduct(string serialNo) constant returns (address,
uint[], uint) {
        return (products[serialNo].producer,
products[serialNo].locationData, products[serialNo].timeStamp);
    }
}
```

Fig. 5.2.1 Provenance Smart Contract Code

B.) Tracking Smart Contract Code

```
// contract to allow supply chain parties to track shipment of goods
and automatically execute payment in tokens
contract Tracking {

    address admin;
    uint[] contractLocation; // array containing lat & long
    uint contractLeadTime; // in seconds
    uint contractPayment; // in tokens
    mapping (string => Shipment) shipments;
    mapping (address => uint) balances;
    mapping (address => uint) totalShipped; // total number of
shipments made
    mapping (address => uint) successShipped; // number of shipments
successfully completed

    struct Shipment {
        string item;
        uint quantity;
        uint[] locationData;
        uint timeStamp;
        address sender;
    }

    // events to display messages when certain transactions are
executed
    event Success(string _message, string trackingNo, uint[]
_locationData, uint _timeStamp, address _sender);
    event Payment(string _message, address _from, address _to, uint
_amount);
    event Failure(string _message);

    // constructor - runs once when contract is deployed
    // determine initial token supply upon contract deployment
    function Tracking(uint _initialTokenSupply) {
        admin = msg.sender;
        balances[admin] = _initialTokenSupply; // all tokens held by
admin initially
    }

    // modifier to allow only admin to execute a function
    modifier onlyAdmin() {
        if (msg.sender != admin) throw;
        _;
    }
```

```
    // function to send tokens from one account to another
    function sendToken(address _from, address _to, uint _amount)
returns (bool success) {
        if (balances[_from] < _amount) {
            Failure('Insufficient funds to send payment');
            return false;
        }
        balances[_from] -= _amount;
        balances[_to] += _amount;
        Payment('Payment sent', _from, _to, _amount);
        return true;
    }

    // function to show token balance of an account
    function getBalance(address _account) constant returns (uint
_balance) {
        return balances[_account];
    }

    // function to recover tokens from an account (can only be done by
admin)
    // in the event that the sendToken function gets abused
    function recoverToken(address _from, uint _amount) onlyAdmin
returns (bool success) {
        if (balances[_from] < _amount) {
            Failure('Insufficient funds for recovery');
            return false;
        }
        balances[_from] -= _amount;
        balances[msg.sender] += _amount;
        Payment('Funds recovered', _from, msg.sender, _amount);
        return true;
    }


    // function to set contract parameters for next leg of shipment
(can only be done by admin)
    function setContractParameters(uint[] _location, uint _leadTime,
uint _payment) onlyAdmin returns (bool success) {
        contractLocation = _location; // set next location that will
receive shipment
        contractLeadTime = _leadTime; // set acceptable lead time for
next leg of shipment
        contractPayment = _payment; // set payment amount for
completing next leg of shipment
        return true;
    }

    // function for party to input details of shipment that was sent
```

```
    function sendShipment(string trackingNo, string _item, uint
_quantity, uint[] _locationData) returns (bool success) {
        shipments[trackingNo].item = _item;
        shipments[trackingNo].quantity = _quantity;
        shipments[trackingNo].locationData = _locationData;
        shipments[trackingNo].timeStamp = block.timestamp;
        shipments[trackingNo].sender = msg.sender;
        totalShipped[msg.sender] += 1;
        Success('Item shipped', trackingNo, _locationData,
block.timestamp, msg.sender);
        return true;
    }

    // function for party to input details of shipment that was
received
    function receiveShipment(string trackingNo, string _item, uint
_quantity, uint[] _locationData) returns (bool success) {

        // check that item and quantity received match item and
quantity shipped
        if (sha3(shipments[trackingNo].item) == sha3(_item) &&
shipments[trackingNo].quantity == _quantity) {
            successShipped[shipments[trackingNo].sender] += 1;
            Success('Item received', trackingNo, _locationData,
block.timestamp, msg.sender);



            // execute payment if item received on time and location
correct
            if (block.timestamp <= shipments[trackingNo].timeStamp +
contractLeadTime && _locationData[0] == contractLocation[0] &&
_locationData[1] == contractLocation[1]) {
                sendToken(admin, shipments[trackingNo].sender,
contractPayment);
            }
            else {
                Failure('Payment not triggered as criteria not met');
            }
            return true;
        }
        else {
            Failure('Error in item/quantity');
            return false;
        }
    }

    // function to remove details of shipment from database (can only
be done by admin)
```

```
    function deleteShipment(string trackingNo) onlyAdmin returns (bool
success) {
        delete shipments[trackingNo];
        return true;
    }

    // function to display details of shipment
    function checkShipment(string trackingNo) constant returns
(string, uint, uint[], uint, address) {
        return (shipments[trackingNo].item,
shipments[trackingNo].quantity, shipments[trackingNo].locationData,
shipments[trackingNo].timeStamp, shipments[trackingNo].sender);
    }

    // function to display number of successfully completed shipments
and total shipments for a party
    function checkSuccess(address _sender) constant returns (uint,
uint) {
        return (successShipped[_sender], totalShipped[_sender]);
    }

    // function to calculate reputation score of a party (percentage
of successfully completed shipments)
    function calculateReputation(address _sender) constant returns
(uint) {
        if (totalShipped[_sender] != 0) {
            return (100 * successShipped[_sender] /
totalShipped[_sender]);
        }
        else {
            return 0;
        }
    }
}
```

Fig. 5.2.2 Tracking Smart Contract Code

C. Reputation Smart Contract Code

```
// contract to store database of supply chain parties and their
reputations
contract Reputation {

    // call the Tracking contract at its deployed address
    // need to include Tracking contract code at the end
    Tracking track =
Tracking(0x0dcd2f752394c41875e259e00bb44fd505297caf);

    address admin;
    mapping (address => Supplier) suppliers;
    address[] suppliersByAddress; // array of all suppliers' accounts

    struct Supplier {
        string name;
        uint phoneNo;
        string cityState;
        string country;
        string goodsType;
        uint reputation;
    }

    // constructor - runs once when contract is deployed
    function Reputation() {
        admin = msg.sender;
    }

    // modifier to allow only admin to execute a function
    modifier onlyAdmin() {
        if (msg.sender != admin) throw;
        _;
    }

     // function for supplier to add their details to database
     function addSupplier(string _name, uint _phoneNo, string
 _cityState, string _country, string _goodsType) returns (bool success)
 {

         // don't overwrite existing entries and ensure name isn't null
         if (bytes(suppliers[msg.sender].name).length == 0 &&
 bytes(_name).length != 0) {
             suppliers[msg.sender].name = _name;
             suppliers[msg.sender].phoneNo = _phoneNo;
             suppliers[msg.sender].cityState = _cityState;
             suppliers[msg.sender].country = _country;
```

```
            suppliers[msg.sender].goodsType = _goodsType;
            suppliers[msg.sender].reputation =
track.calculateReputation(msg.sender);
            suppliersByAddress.push(msg.sender);
            return true;
        }
        else {
            return false; // either entry already exists or name
entered was null
        }
    }

    // function to remove supplier from database (can only be done by
admin)
    function removeSupplier(address _supplier) onlyAdmin returns (bool
success) {
        delete suppliers[_supplier];

        // delete entry from array and shorten array
        for (uint i = 0; i < suppliersByAddress.length; i++) {
            if (suppliersByAddress[i] == _supplier) {
                for (uint index = i; index < suppliersByAddress.length
- 1; index++) {
                    suppliersByAddress[index] =
suppliersByAddress[index + 1];
                }
                delete suppliersByAddress[suppliersByAddress.length -
1];
                suppliersByAddress.length--;
            }
        }
        return true;
    }

   // function to display details of supplier
   function findSupplier(address _supplier) constant returns (string,
uint, string, string, string, uint) {
       return (suppliers[_supplier].name,
suppliers[_supplier].phoneNo, suppliers[_supplier].cityState,
suppliers[_supplier].country, suppliers[_supplier].goodsType,
suppliers[_supplier].reputation);
   }

   // function to display all suppliers' accounts in database
   function allSuppliers() constant returns (address[]) {
       return suppliersByAddress;
   }
```

```
    // function to search for suppliers by type of goods
    function filterByGoodsType(string _goodsType) constant returns
(address[]) {
        address[] memory filteredGoods = new
address[](suppliersByAddress.length);
        for (uint i = 0; i < suppliersByAddress.length; i++) {
            if (sha3(suppliers[suppliersByAddress[i]].goodsType) ==
sha3(_goodsType)) {
                filteredGoods[i] = suppliersByAddress[i];
            }
        }
        return filteredGoods;
    }

    // function to search for suppliers by reputation (returns those
with same or higher reputation)
    function filterByReputation(uint _reputation) constant returns
(address[]) {
        address[] memory filteredRep = new
address[](suppliersByAddress.length);
        for (uint i = 0; i < suppliersByAddress.length; i++) {
            if (suppliers[suppliersByAddress[i]].reputation >=
_reputation) {
                filteredRep[i] = suppliersByAddress[i];
            }
        }
        return filteredRep;
    }

    // function to display reputation of supplier (calls Tracking
 contract)
    function checkReputation(address _supplier) constant returns
(uint) {
        return track.calculateReputation(_supplier);
    }

    // function to update reputations of all suppliers (calls Tracking
 contract)
    // can only be done by admin
    function updateReputations() onlyAdmin returns (bool success) {
        for (uint i = 0; i < suppliersByAddress.length; i++) {
            suppliers[suppliersByAddress[i]].reputation =
track.calculateReputation(suppliersByAddress[i]);
        }
        return true;
    }
}
```

Fig. 5.2.3 Reputation Smart Contract

Chapter 6

Results & Discussion

6.1 Results with respect to impact of packaging variables on total packaging cost

Table 6.1.1 Accuracies of Machine learning models implemented on packaging variables

Method	Accuracy
Multiple Linear Regression	83.51%
Polynomial Regression	80.78%
Random Forest Regression	64.10%
Decision Tree Regression	53.56%

Linear regression is found to have achieved the highest accuracy (least error) & thus this is the model we are going to consider.

In our MLR model we predicted the importance of each of the variables and achieved the following result:

```
In [20]:  1 Coefficient_Weights

Out[20]: {'Material Cost': 14.812799564537796,
          'Storage & Handling Cost': 17.63661830964852,
          'Labour Cost': 20.48975862326298,
          'Accessory Cost': 15.253467496318567,
          'Transport Cost': 15.808884040362651,
          'Cost of Return': 15.998471965869488}
```

Fig. 6.1.1 Coefficients of Weights of Packaging Variables

From the above coefficients we can see that different factors contribute with different impacts to the total cost.

In this particular case labour cost was found to have the highest impact on the total cost of packaging.

6.2 Results with respect to Sentimental Analysis

Table 6.2.1 Accuracies of Machine learning models implemented on Sentiment Analysis

Model	Accuracy
Multinomial Naive bayes	0.85
Bernoullis Naive bayes	0.8
Logistic regression	0.93

Based on the reviews collected for the television, sentiments of 90.1% were positive, and 9.9% were negative. This basically signifies that in the majority of cases, the customers are satisfied with the package provided and the package evaluation ends on a positive note. Whatever might be the accuracy there is always room for improvement so here word clouds were generated

which represent both negative and positive words most frequently used in the context. Negative words of large size represent the most occurring problem and the need for rectification of it.

The below chart represents the word cloud of positive and negative words:

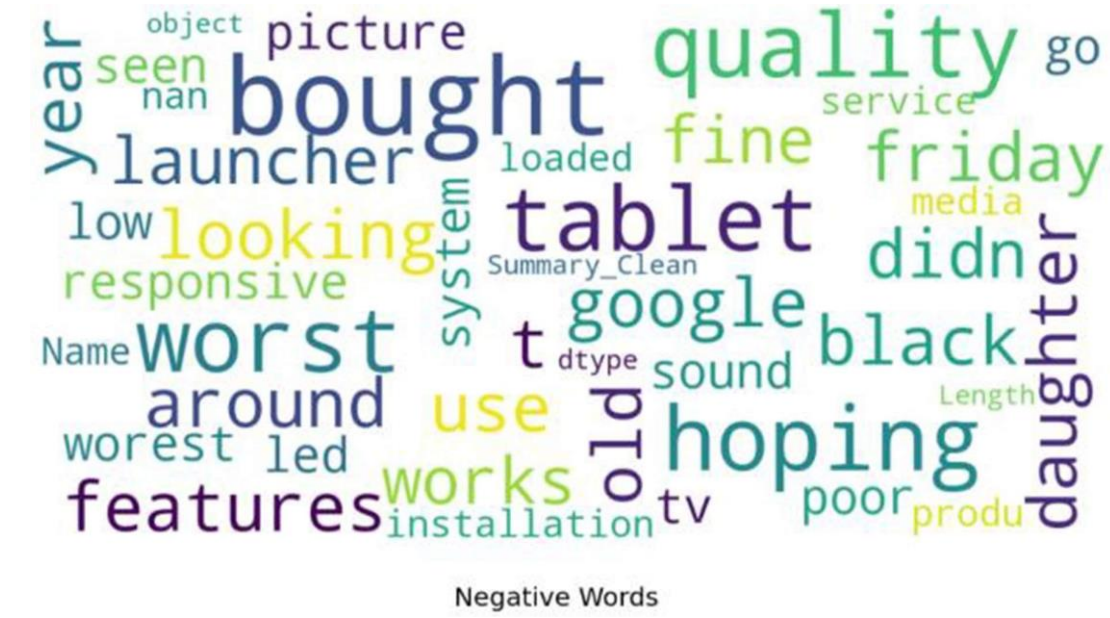

Fig. 6.2.1 Word Cloud of Negative Words

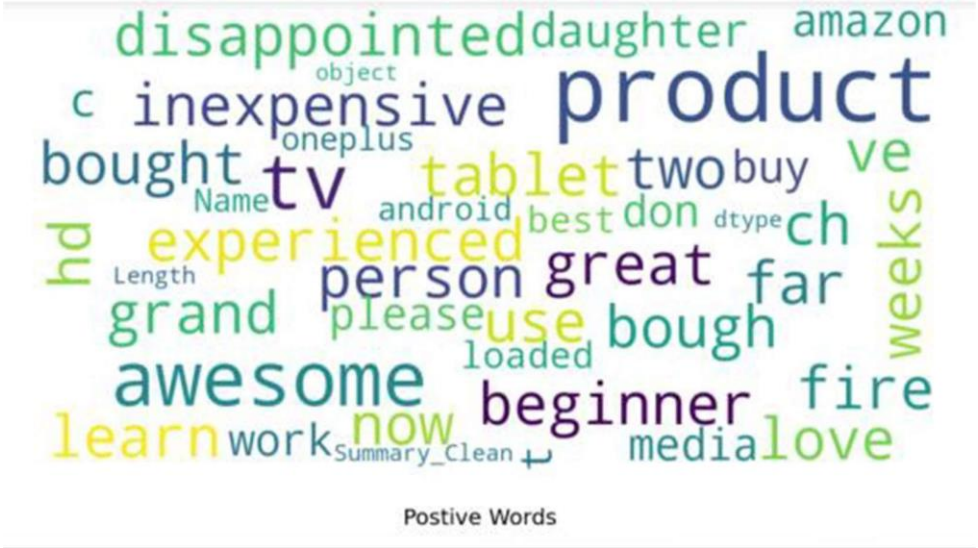

Fig. 6.2.2 Word Cloud of Positive Words

6.3 Results with respect to Smart contracts

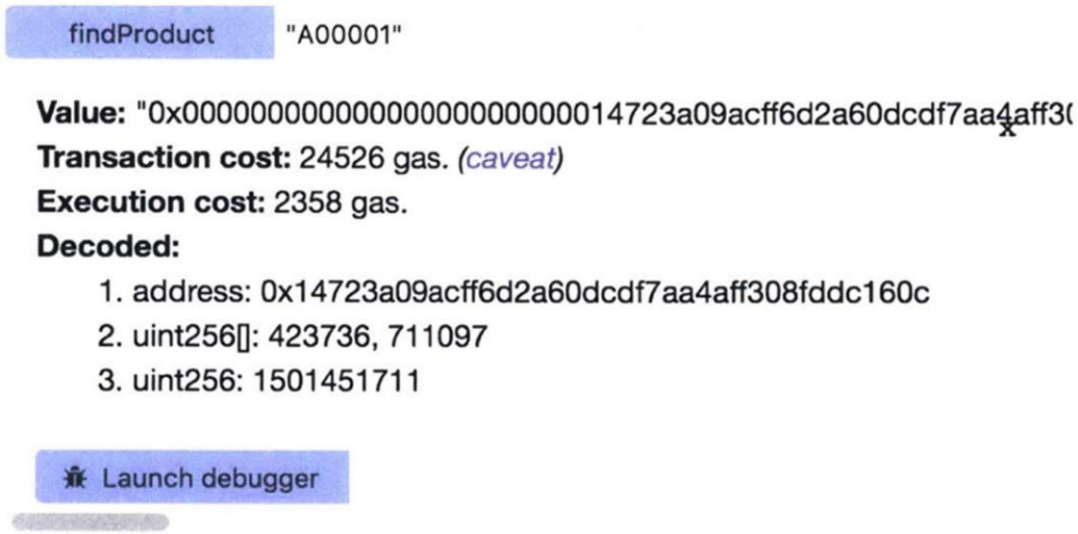

Fig. 6.3.1 Code for function findProduct

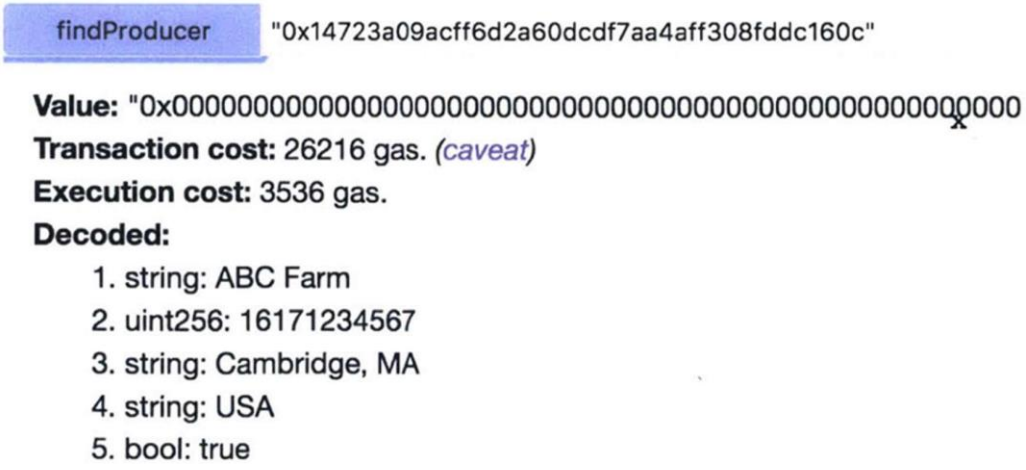

Fig. 6.3.2 Code for findProducer

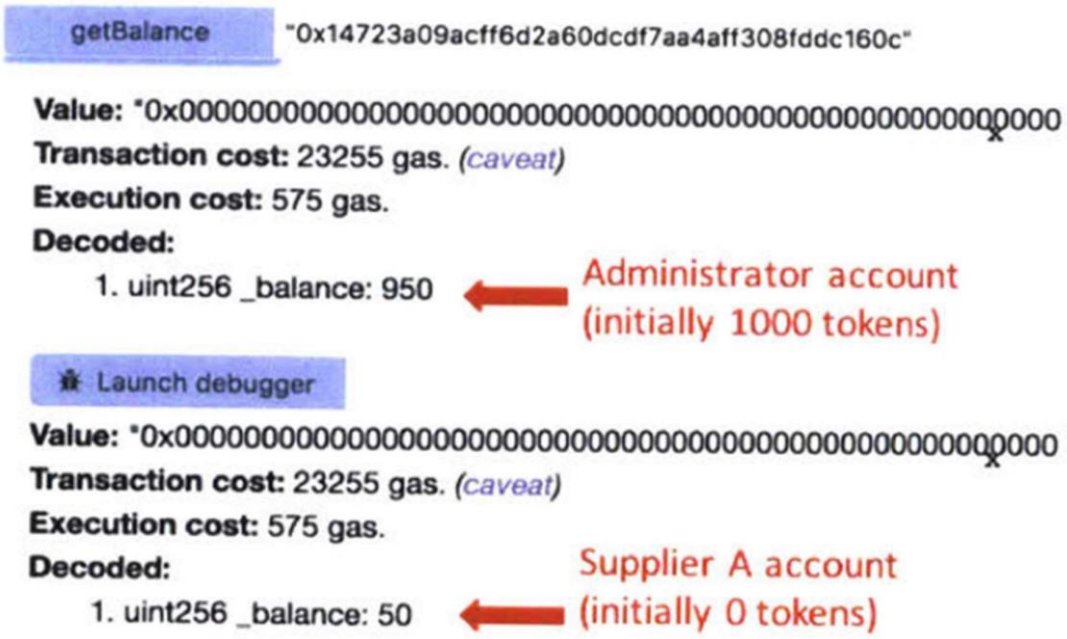

Fig. 6.3.3 Code for getBalance & launchDebugger

Name	Public Address	Goods Type	Reputation
ABC Farm	0x14723a09acff6d2a60dcdf7aa4aff308fddc160c	Batteries	75
DEF Company	0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db	Wafers	50
GHI Company	0x583031d1113ad414f02576bd6afabfb302140225	LCD	100
JKL Company	0xdd870fa1b7c4700f2bd7f44238821c26f7392148	Batteries	0

Fig. 6.3.4 Input Data

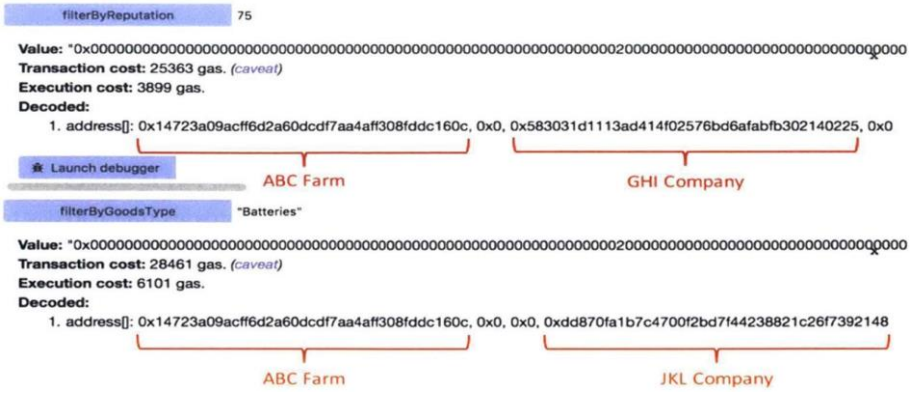

Fig. 6.3.5 Code for filterbyReputation and filterbyGoodsType

Chapter 7

Conclusion & References

7.1 Conclusion

From the above machine learning models, we can conclude that in the case of the electronics industry, labor costs are having the greatest impact on total packaging cost among the six variables we have considered. This could be the case because in the electronics industry we have a lot of fragile items which need to be handled with a lot of care and also require extra efforts by workers to pack and wrap the materials properly.
The above results were generated using preliminary data sources for the electronics sector in particular. In the future we will gather data from numerous leading industry experts which will lead to a more accurate dataset with which to work. We also aim to extend our work to include different sectors like fashion, retail etc

Package performance was evaluated from the customer reviews instead of physical testing and as a result, the evaluation process was found to be more effective. For the television taken into consideration, out of the total reviews, '90.1%' of reviews depicted a positive opinion and it can be framed that the result of the package evaluation was on a positive note. The overall percentage of failure can be used to ensure the assurance level of the current package design and it also helps to identify any seasonal issues. Moreover, a word cloud of negative reviews can be used as an indicator of the most problematic packaging areas.

Smart contracts code was run in the solidity and sample functions like Findproduct , FindProducer , getBalance , Launchdebugger , filterbyReputation and filterbyGoodsType were validated and tested.

7.2 References

• Visser E., 2002, Packaging on the web: an underused resource. Design Management Journal, 62-67.

• Kalakota R., Whinston A.B., 1997, Electronic Commerce: a Manager's Guide, Addison-Wesley, Reading, MA.

• Huang K.L., Rust C., Press M., 2009, Packaging design for e-commerce: identifying new challenges and opportunities for online packaging, College of Digital Design, Visual Communication Design Graduate School of Digital Content and Animation.

• Dyllick T., 1989, Ecological marketing strategy for Toni yoghurts in Switzerland, Journal of Business Ethics, 8(8), 657-662.

• Hellström D., Saghir M., 2006, Packaging and logistics interactions in retail supply chains, Packaging Technology and Science, 20(3) 197-216.

• Twede D., 1992, The process of packaging logistical innovation, Journal of Business Logistics, 13(1) 69-94.

• Prendergast G., Pitt L., 1996, Packaging, marketing, logistics and the environment: are there trade-offs? International Journal of Physical Distribution & Logistics Management, 26(6), 60-72.

• Manzini R., Gamberi M., Gebennini E., Regattieri A. 2008, An integrated approach to the design and management of a supply
chain. International Journal of Advanced Manufacturing Technology, 37(5-6), 625-640.

• Manzini R., Ferrari E., Gamberi M., Persona A., Regattieri A. 2005. Simulation performance in the optimisation of the supply chain. Journal of Manufacturing Technology Management, 16(2), 127- 144.

• Korzeniowski A., Jasiczak J., 2005, Accomplishment of selected package functions in e-commerce, LogForum,, 1, 1-7.

● Sarkis J., Meade L.M., Talluri S., 2004, E-logistics and the natural environment, Supply Chain Management: an International Journal, 9(4), 303-312.

● Lepine J, Rouillard V, Sek M. Review paper on-road vehicle vibration simulation for packaging testing purposes. Packag Technol Sci: Int J. 2015;28(8):672-682.

● Young DE. Testing and evaluation of transport packaging: a view to the future. Packag Technol Sci: Int J. 2000;13(1):3-6.

● Shiva Esfahanian, Euihark Lee, A novel packaging evaluation method using sentiment analysis of customer reviews: