

Data Extraction from Social Media for Sentiment Analysis in order to Predict Sales and Correlated Items for Fashion Industry

C.S Kumanayaka

169315X

Faculty of Information Technology

University of Moratuwa

December 2018

Data Extraction from Social Media for Sentiment Analysis in order to Predict Sales and Correlated Items for Fashion Industry

C.S Kumanayaka

169315X

Dissertation submitted to the Faculty of Information Technology, University of
Moratuwa, Sri Lanka for the partial fulfillment of the requirements of the Honors
Degree of Bachelor of Science in Information Technology

December 2018

Declaration

I declared that this thesis is written by myself and it has not submitted by other institution of education, degree or diploma of any other university. Information retrieved from unpublished and published work listed in the reference area.

Name of Student

Signature of Student

Date:

Supervised by

Name of Supervisor(s)

Signature of Supervisor(s)

Date:

Acknowledgment

Initially I am thankful to my supervisor Mr. Saminda Premaratne the valuable advice and supervision. I like to give my gratitude to him for his important time for long discussions throughout my research. His rightful guidance paved the way to achieve priceless achievements

Special thanks should also go to Mr Sankha Perera for the advice and guidance provided from a Statistical point of view. I am also thankful for Mr Prabhath Gunawardane for his valuable guidance and advice in Data Science. Gratitude should also go to Ms.Ayesha Kasthuriarachchi for the statistical advice and guidance.

Also, I should be thankful to all the lecturers about the assistance and guidance provided in carrying out the project. Finally, I should also be grateful to all my batch mates and all the other parties who helped us with their valuable ideas to proceed with my research successfully.

Abstract

These days' Social media is being very popular for marketing. One of the most popular social media platforms is Facebook. Most of the Fashion brand have Facebook pages. Consumer expresses their feeling using comments and emotional buttons. The user interacts with the brand page using post, like, share comments. The analyzed data can give support to decision makers to the evaluation of the customer's feedback, identify potential customers and predict sales item for the upcoming month and predict correlated items. The purpose of this paper is to explain how to extract and prepare data collected on Facebook to perform sentiment Analysis.

Table of Content

Declaration	ii
Acknowledgement	iii
Abstract	iv
Table of Content	v
List of Figures	vi
Abbreviations	viii
Chapter 1	1
Introduction	1
SQL Server Data Tools (SSDT) for Visual Studio	4
Chapter 2	5
Literature Review	5
Chapter 3	13
Technologies Used for the Tool	13
Chapter 4	16
Analysis and Design	16
4.1 Social media data extraction	17
4.2 Preprocessing of Text data	17
4.3 Feature Extraction using Text Data	17
4.5 Sentiment Analysis (Sentiment model building and training)	18
4.6 Predict Sales using Sentiment and Semantic (Sales model building and training)	19
4.7 Test and evaluation	19
4.8 Identify correlated item	20
4.9 Live Dashboard (Correlated item, sales prediction and insight about items) and mobile application	20
Chapter 5	21
Implementation	21
5.1 Data Collection	21
5.2 Data Preprocessing	22
5.3 Feature Extraction using Text Data	28
5.4 Split training and testing data	32

5.5 Sentiment Analysis (Sentiment model building and training).....	33
5.6 Predict Sales using Sentiment and Semantic (Sales model building and training)	36
5.7 Predicting Sales item and correlated item	38
5.8 Live Dashboard (Correlated item, sales prediction and insight about items) and mobile application	39
Chapter 6.....	41
Evaluation	41
6.1 Accuracy Check the textblob library	41
6.2 Accuracy of Sentiment model	41
6.3 Accuracy of Sales model	42
Chapter 7.....	43
Conclusion and further work	43
7. 1 Overall Achievement.....	43
7. 2 Achievement of each objective.....	43
7. 3 Problem encountered	44
7.4 Limitation	44
7.5 Further work	44
References.....	45

List of Figures

Figure1.1 Research overview diagram Terminology	04
Figure 2.1 Facebook Terminology [2]	08
Figure 2.2 Part of the dictionary of emoticons and emoji's [2]	08
Figure 2.3 System Design	09
Figure 2.4 Performance Evaluation	11
Figure 2.5 Social Media is important to my business	12
Figure 2.6 Weekly time commitment for social media marketing	12
Figure 4.1 System design	17
Figure 4.2 Train/Text Split	19
Figure 5.1 Raw data from 'Thefashionworks' facebook page	22
Figure 5.2. Data cleaning	23
Figure 5.3 Lower casing	24
Figure 5.3 Remove Punctuation	24
Figure 5.4 Remove Stopwords	25
Figure 5.5 Remove Common Word	26
Figure 5.6 Remove Rare Words	27
Figure 5.7 Spelling Correction	27
Figure 5.8 Tokenization	28
Figure 5.9 Stemming	28
Figure 5.10 Lemmatization	29
Figure 5.11 Number of words in the comments	30
Figure 5.12 Number of characters in the comments	30
Figure 5.13 Average word length in the comments	31
Figure 5.14 count number of stop word from comments	31
Figure 5.15 Number of special character word from comments	32
Figure 5.16 generate number of numeric from comments	32
Figure 5.17 Generate number of upper case word in the comments	33
Figure 5.18 splitting the training and testing data set.	34
Figure 5.19 detecting the sentiment using the textblob library	34
Figure 5.20 python code for splitting and setting dependent variable	35
Figure 5.21 the python code for feature matric creation.	35

Figure 5.22 python code fitting and training model	36
Figure 5.23 Predict the test data using model	36
Figure 5.24 plotting the model	36
Figure 5.25 plot sentiment model	37
Figure 5.26 Python scripts for dependent/independent variable setting and fitting the model	38
Figure 5.27 some of sales prediction for testing data set	38
Figure 5.28 plotting the sales model	39
Figure 5.29 exiting item category	40
Figure 5.30 check correlation - python script and results.	41
Figure 5.31 Some charts of dashboard	42
Figure 6.1 Checking Accuracy	43
Figure 6.2 Shows the python scrip for generate accuracy of sales model	44

Abbreviations

FB	Facebook
ETL	Extraction transformation load
SSIS	SQL Server Integration service
SSRS	SQL Server Reporting service
BI	Business Intelligence

Chapter 1

Introduction

Social media can combine marketing and Information Technology(IT). A number of people who interacts social media is increasing day by day. The amount of data produced in social media is also increasing massively. It is a big challenge to analyze the social media data product wise. Management staff had to be more warn and alert about the negative comments and posts because that can be harmful to the image of a brand or product.

The fashion industry is very dynamic. Day by day new thing is coming to the industry. People choices are changing drastically. Due to this reason, fashion Product should align with these changes.

In industry, usually consumer feedback is complex. Currently, it takes a lot of time and effort in order to analyze and data cleaning. But fashion product review analyze is time-consuming work and it takes more labors.

Most of the time social media market analysis is doing by manually. There are no well develop a mechanism to analyze subjective social media data. When we compare with other product people, give more review to fashion product through social media. Fashion product review is more subjective than other product. Because of this, the complexity of fashion product review is higher than other product and difficult to analyze.

Recent literature points out that the term Sentiment Analysis was perhaps first used by Nasukawa and Yi[3]. Sentiment Analysis, also called Opinion Mining, is the process of collecting opinions, emotions or even attitudes from a text analysis [4]. The purpose is to understand and determine a communication through the contextual polarity in a text or reaction in a Social Media platform [5].

Aim

The main aims of the research to reduce the market analysis duration and provide higher managers the feedback in order to align their business operation accordingly and identified disadvantages of the existing techniques or systems, aligned to address the identified drawbacks.

Objective to achieve the aim

- Research includes developing a live dashboard, is totally accessible 24/7 so management can be aware of what is happening to be and predict about the item's sales, identify Co-related items, predict sales item for an upcoming month and identification of the feeling is positive, negative and act at the moment.
- The proposed solution identifies upcoming month sales and correlated items based on sentimental and semantic analysis.
- Gain of the successful research which can be led to catering many different areas and situations.
- Analyze the comments, emojis and review for upcoming items.
- Predict upcoming sales through qualitative and quantitative social media data and previous months' sales.
- Identify the correlated item, that can be a sale with top selling items category.
- It will make a cost-effective solution for the fashion Industry
- Real-time dashboard and mobile application for top management.
- Provide a well develop a mechanism to analyze subjective social media data.

Solution

Most of the Fashion clothes shops publish their coming month items through social media. Extracted Facebook analytic used for feature extraction, data preprocessing and data modeling.

This solution consists of two models as the sentiment score model and sales score model. In this sentiment, the model is to generate the score for all the comments. Here, extracted polarity as it indicates the sentiment like if the sentiment value near to 1, it means a positive sentiment and values near to -1, it means negative. For the second model is to predict sales for the upcoming month as featuring previous Facebook analytics and previous sales counts. Next part of research is identifying the correlated item category that can be a sale with upcoming items.

The implemented live dashboard provides valuable insight for top management. Using this information in a correct manner can be used in organizations to gain advantages. Bring the real-time, can determine if the identification of the feeling is positive, negative and act at the moment.

Sentiment Analysis and semantic Analysis can provide useful information about

customers. As is a live dashboard, is totally accessible 24/7 so management can be aware of what is happening to be and predict about the item's sales. This analysis can be used for better marketing strategies.

What is expected to be achieved through this research depicted in figure 1.1.

Research Overview Diagram

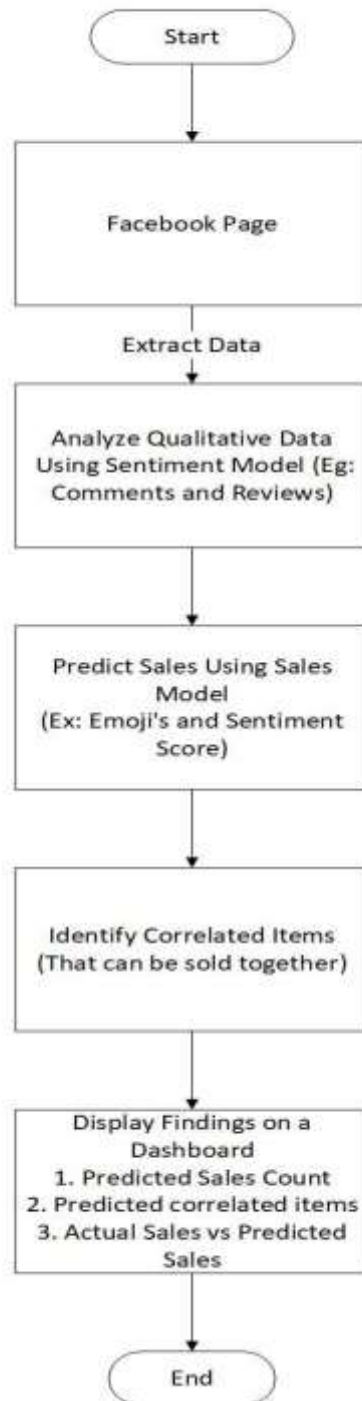


Figure1.1 Research overview diagram

Below shown the technologies used to achieve the research.

Facebook for the developer (Graph API Explorer)

The Graph API is used to extract data into and out of the FB (Facebook) platform.

Visual Studio Python

Python Tools for Visual Studio (PTVS) is used to feature extraction and text preprocessing, text processing, model building and model evaluation.

Power BI

Power BI used to generate dashboards and dataset that allow to exploring data with modern visualizations and spot trends.

SQL Server Integration Tool

Integration Services used to integration data and transform data from Facebook data to relational database and then load the data into one or more destinations.

SQL Server Data Tools (SSDT) for Visual Studio

Sql server Data tool is essential for sql server integration service(SSIS)

SQL Server Management Studio

SQL Server Management Studio (SSMS) is used to store the data.

Chapter 2

Literature Review

The purpose of “Data extraction and preparation to perform a sentiment analysis using open source tools” research was creating a personalized tool to extract and prepare the data from Facebook of fashion brand page. After this research project, the ultimate motivation of the research used a sentiment analysis tool to get the sentiments from Facebook comment towards the brand. Another objective was to collect the reactions created by some posts about products and with the help of other teams understand why producers with a high expectation of being a best seller were not. With marketing team is to understand if people who ask information about prices, buy the product and compare with competitors. [2]

Antonio Teixeira and Raul M. S. Laureano [2] defined a five phrases to Predicting and Visualizing Consumer Sentiments in Online Social Media.

The whole process is complex and comprises five steps [2]:

1. Data collection: In this step, they collected the data from Social Media Platforms. The data was in an unstructured form.
2. Data preparation: Here they cleaned all non-textual contents from the data they collected.
3. Sentiment detection: Then, they examined the opinions and thoughts collected. Only the opinions, beliefs, views are maintained.
4. Sentiment classification: In this step, they classified the sentiments gathered in the data collected as positive, negative, neutral.
5. Presentation of output: Here the main goal was converting the information in "readable way"[2]

Technologies used for Research

- Facebook Graph API
- Rest FB

These are some finding throughout this research.

- If the system wanted to get the total number of post of a page, it has blocked at the same time. When the API was called a big number of consecutive time, it got blocked.
- Another issue they had is the different languages used in the comments. They had Spanish, Italian, Portuguese, French, English and Arabic. So, they used Google Translation API to detect the language used in the comment. That was a great technique to social media data analyze. Because their product was internationally branded, so, they should analyze different languages used in the comments.
- Some comments mention of the friend's name. The system should be including proper cleaning part to cleanser emoticons and separate the real comments from the others. They were investigating if using Stanford Named Entity.

The general data they stored in the relational data base was:

- i) Post identification
- ii): The unique identifier of the post;
- iii) Post message: The message that was in the post;
- iv) Number of shares: Number of times the post was shared;
- v) Number of total reactions types: Total number of reactions, and total

number of reactions type;

vi) Post product: Product announced in the post;

vii) Post page: Page where the post was made;

viii) Post date: Date the post was made;

ix) Person: Name of the person who made the reaction or the comment.

x) Comments: Comments made in the post. [2]

They categorized Facebook terminology. (Figure 2.1)

<i>User</i>	<i>Activity</i>	<i>Facebook Term</i>
1	Status update	Post, Status; Status post
2	Agree with status update of user 1	Like; Love, Haha; Wow
2	Disagree with status update of user 1	Sad; Angry
2	Make a comment	Comment
2	Reply a comment	Reply
2	Share User 1 status update	Share

Figure 2.1 Facebook Terminology [2]

Emoticons and emojis are categorized as bellows. (Figure 2.2)

<i>Emoticon</i>	<i>Polarity</i>
:-) :) :o) :] :3 :c)	Positive
😊, ❤️, ♥️	Extremely-Positive
:D C:	Extremely-Positive
:-(:(:c :[Negative
D8 D; D= DX v.v	Extremely-Negative
😞, ♥️	Extremely-Negative
:	Neutral

Figure 2.2 Part of the dictionary of emoticons and emojis [2]

The purpose of “A semantic graph-based approach on interest extraction from user-generated texts in social media” was Analyzing the Micro-blogs and social networking websites texts to find the main topics mentioned in them is a fine method for targeted marketing. Targeted marketing involves identifying potential clients who might be interested in particular products or services and marketing them to these clients.[1]

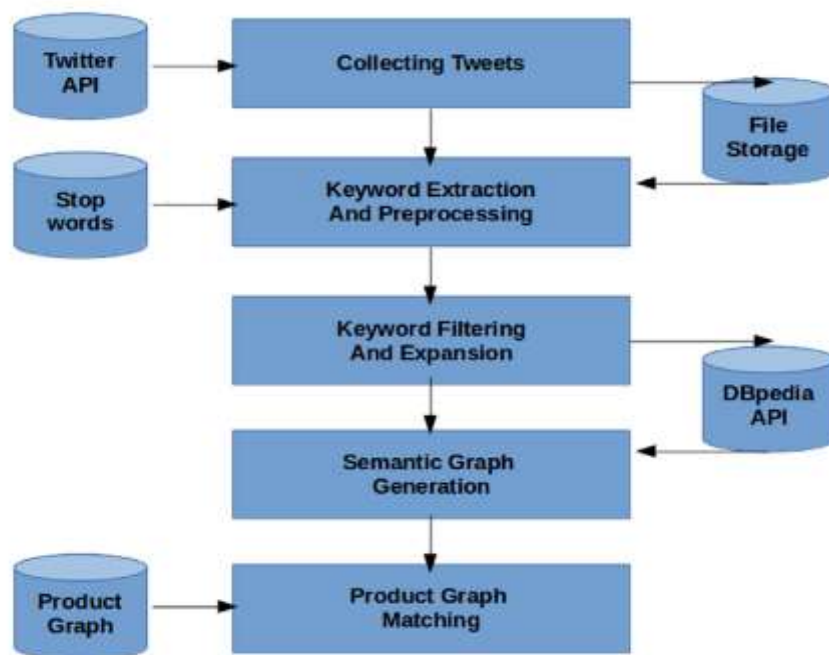


Figure 2.3 System Design

Technologies/Algorithm used for Research

- Rapid Automatic Keyword Extraction algorithm. (Rake)
- Twitter API
- Tweepy (python library)
- SPARQL query

Steps

- Extract a set of tweets from Twitter.
- Identify Twitter API credentials for authentication.
- To communication with Twitter API ,Tweepy (python library) has been used.
- Using a single API request, two hundred tweets can be extracted. So, there was a loop for multiple request. But still, they were able to collect 3240 tweets only.
- Noun phrases raked using term frequency.
- There were several preprocessing steps. They were removed white spaces, replace the same character using a single character, replace hash tags and etc.
- Convert keyword into the format that matched with page reference (DBpedia) after the reading.
- Replaced Twitter user names with a real name.
- To communicate DBpedia database used SPARQL query.

Below display the two methods used for keyword extraction.

1. First of all, they Identified the noun phrases (in the sentence) and after that ranked them using term frequency. Very low frequencies were eliminated.
2. The second method was RAKE algorithm. It is a very popular keyword extraction algorithm in text classification and topic extraction.

These are some finding throughout this research.

- DBpedia has good performance than services online.
- The Research considered English Tweets Only since scripts of text processing were only compatible with English.
- What they found from the research is, interest extraction or Semantic graph-based approach for topic worked gave better performance than traditional methods (topic extraction) like RAKE.

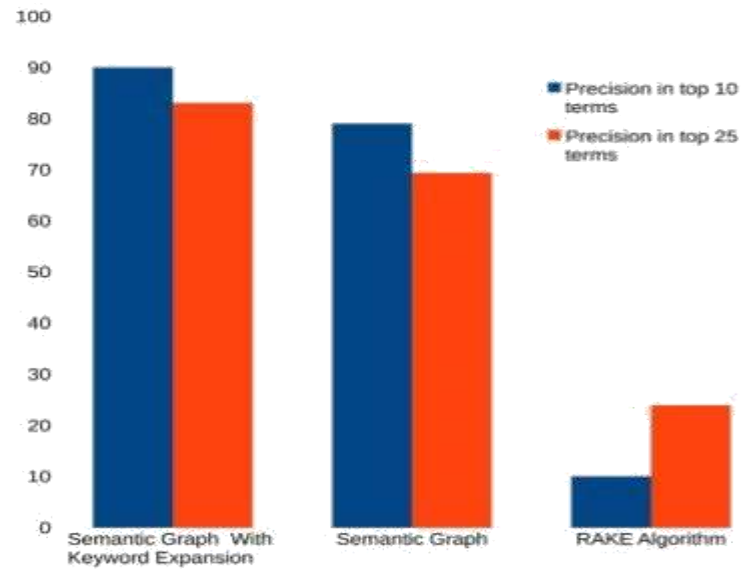


Figure 2.4 Performance Evaluation

MICHAELA STELZNER wrote a report about how marketers are using social media to grow their business. 90% of marketers said that social media was important to their businesses. 26% of companies can measure social activity. This is slightly down from our 2015 findings, where 58% strongly agreed and 34% agreed. [8]

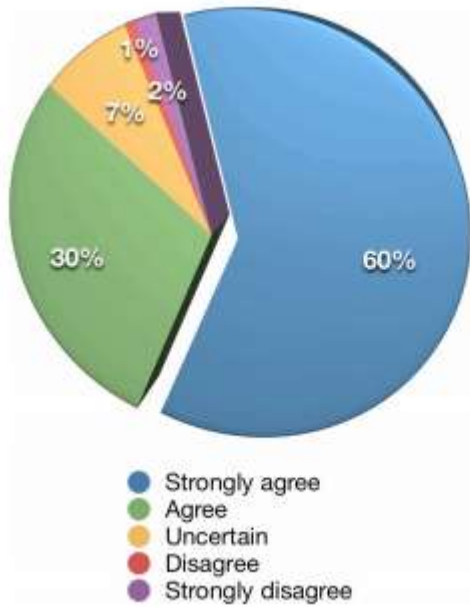


Figure 2.5 Social Media is important to my business

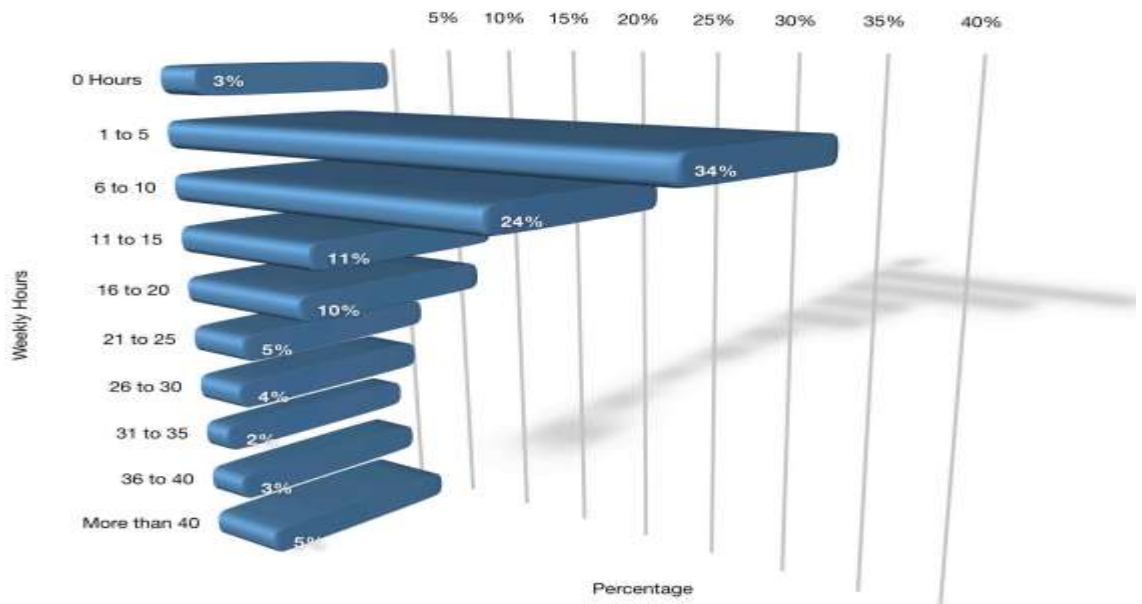


Figure 2.6 Weekly time commitment to social media marketing

A significant 63% of marketers are using social media for 6 hours or more and 39% for 11 or more hours weekly. [8]

Major findings

Here is some summary of their valuable findings:

- Facebook and YouTube are the most popular for future plans. At least 63% of brand marketers planning to increase their sales using social networks.
- Snapchat is also growing social network: Only 5% of brand marketers are have a Snapchat account, other 16% are planning to increasing their Snapchat activities and the other 28% of brand marketers keen about Snapchat.
- Facebook is the most popular social network for brand marketers. When needed to choose their most popular platform, 55% of brand marketers select Facebook, followed by LinkedIn at 18%. Plus, 67% of brand marketers want to increase their activities regarding Facebook marketing.
- Most of the marketers are not sure regarding their Facebook marketing: A 40% of marketers didn't know about the Facebook traffic has and 35% of marketers not sure about Facebook marketing efficiently.
- Facebook ads: A 86% of marketers daily using Facebook advertisements, only 18% use Twitter advertisements.
- Engagement and tactics and are best areas for marketers who want to master: At least 90% of marketers keen about most efficient social tactics and the most comfortable ways to engage with their audience using social media.
- More than half of sales and marketers have been using social medias for improve their sales at least 2 years.

Chapter 3

Technologies Used for the Tool

Graph API Explorer

The Graph API is the way to extract data from the Facebook platform and push data to the Facebook platform. It is a low-level API based on HTTP. The application can use to code for retrieving query data, share stories, administrative advertisement, upload videos/photos, and act a large number of tasks.

The main asset of this project is data from the Facebook platform. One of biggest advantages of Facebook's Graph API against others is that it doesn't need use any special code structure to get most of the information from their API and just have to follow the regular process to consume a JSON response.[7]

Main advantages of Facebook's Graph API is the "Graph API Explorer" .It creates a request to "console" in big documentation , because of that it easy to get an idea of several criteria.

Visual Studio Python

Python Tools for Visual Studio (PTVS) is a plug-in .it is used for versions of Visual Studio up to VS 2015.it is supporting for programming in Python. It provides data cleaning, debugging, data preprocessing, profiling, cluster and more.

Python was used for future extraction, text preprocessing and text processing since Python has good predefined libraries.

Power BI

Power BI pulls data from the Facebook account and generates a dashboard and dataset that allow to exploring the data with modern visualizations. It is easy to create and share reports. Use your Facebook page analytics in Power BI to spot trends such as find out who the most engaged fans are, which posts have been the most shared or how trends have changed over time.

Power BI direct query can be generating a real-time dashboard. That's the main reason for selecting it as a reporting tool. Power BI mobile layout was generated for the mobile users as well. Top Management wanted to share real-time sales update in the client meeting; Power BI provides desktop and mobile layout.

Power bi desktop version was used to dashboard development. It was free with office 365 licensee. Power BI is a Microsoft product, but it isn't exclusive to Microsoft systems. It can be used to retrieved data from various data sources. If something went wrong, Managers can quickly drill into the visualization (dashboards) and look at the data that analyze it. Otherwise, they need to learn a complex query language. Here top management can get a quick data analysis by typing simple sentences.

SQL Server Data Tools (SSDT) for Visual Studio and SQL Server Integration Service

SQL Server Data Tools is a modern development tool.It helps to build Reporting Services (RS) reports ,SQL Server relational databases, Azure SQL databases, Integration Services (IS) packages and Analysis Services (AS) data models. With SSDT, developers can deploy and design any SQL Server content type easily like developing an application in Visual Studio.

SSIS provides data transformation functionality. Analyzed data transformation is the main task to achieve accurate insights. The social media data file can be huge, so performance will be slow. So we should consider parallel data processing. SSIS can be loaded in parallel to many varied destinations. There is a free version that can be installed with SQL server express edition.

SQL Server Management Studio

SQL Server Management Studio (SSMS) is a Microsoft software application. It first launched in 2005. That is used for administering, managing configuring and all appurtenant within Microsoft SQL Server. The tool contains both script graphical and editors who work with features and objects of the server. SISIS express is a totally free version, and it has high security as data a repository.

Analysis and Design

System Design Diagram

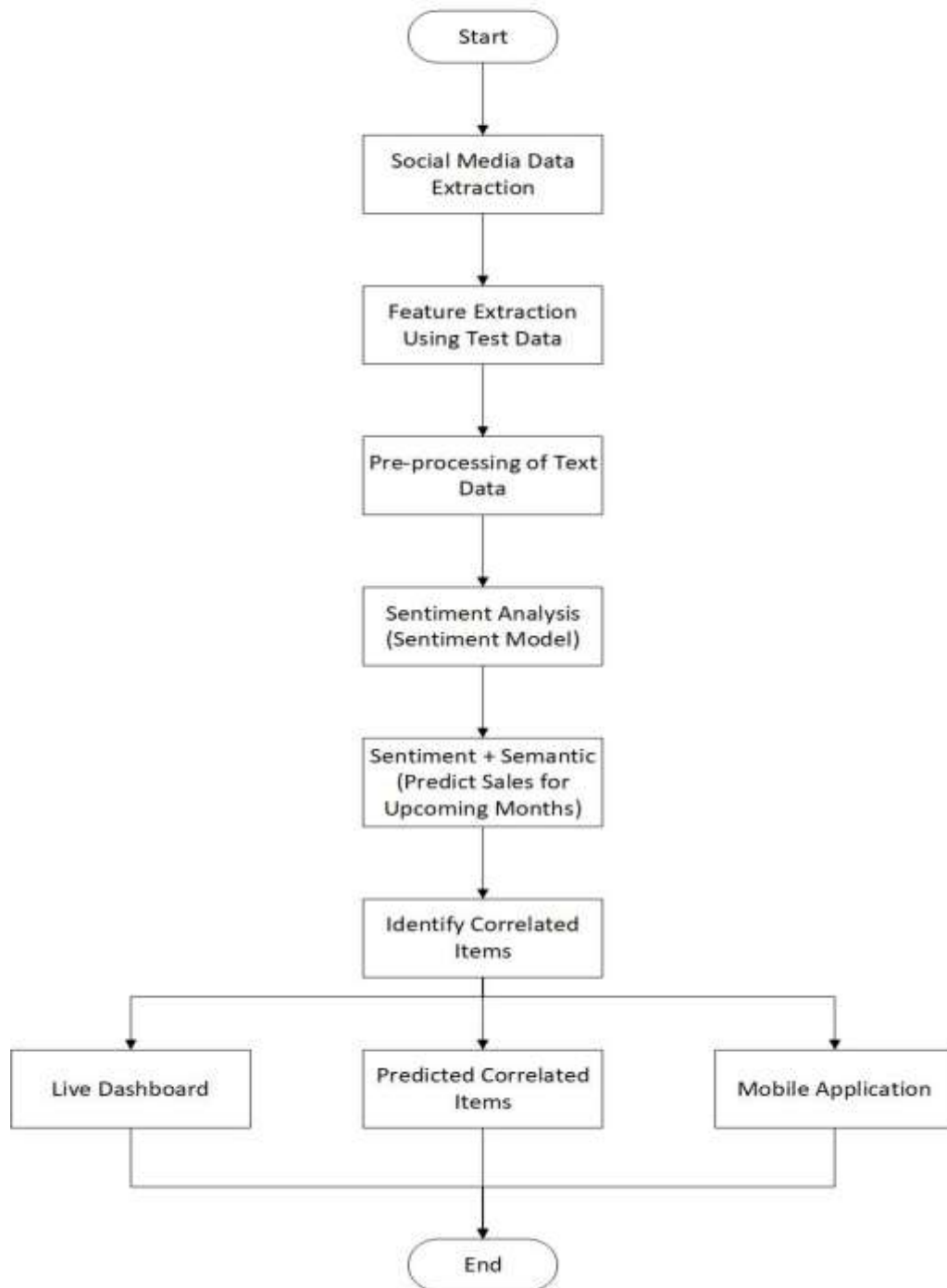


Figure 4.1 System design

4.1 Social media data extraction

The purpose is to extract data from the social media platform. “Prettytheoutlet” is the face book fan page for a clothing store . These clothes shops publish their coming month items through their fan page. one week after publishing Facebook data extracted from Graph API.

4.2 Preprocessing of Text data

Extracted Data is used as input for data preprocessing. Before dividing into text and train data , my first step was cleaning the data to extract better features. achieved this by doing some of the main pre-processing steps on the data. All the English comment only considered to the text analyzed and all the Sinhala comment filtered out form preprocessing.

- Filter the English comments and reviews
- Lower casing
- Punctuation removal
- Stopwords removal
- Frequent words removal
- Rare words removal
- Spelling correction
- Tokenization
- Stemming
- Lemmatization

4.3 Feature Extraction using Text Data

Preprocessed Data is used as input for this part. In this section, explained multiple feature extraction methods, begin with some basic techniques which guide into advanced Natural Language Processing techniques.

Below Step followed for feature extraction using text data

- Number of words

- Number of characters
- Average word length
- Number of stopwords
- Number of special characters
- Number of numerics
- Number of uppercase words

4.4 Split training and testing data

The Data set is split into training data and test data. The training set contains a known output and the model learns on this data in order to be generalized to other data later on. The test dataset (or subset) used to test our model's prediction on this subset. Test data size was 20 % of the whole data set. Rest of them were training data.

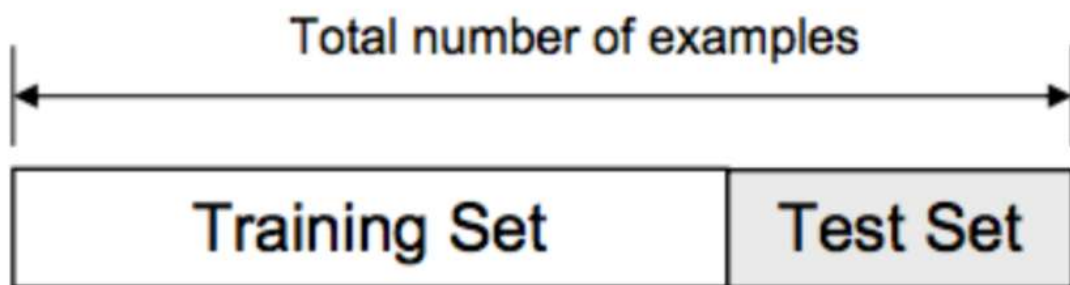


Figure 4.2 Train/Text Split

4.5 Sentiment Analysis (Sentiment model building and training)

The sentiment was generated by textblob library for each and every comments. Extracted polarity as it indicates the sentiment as value nearer to 1 means a positive sentiment and values nearer to -1 means a negative sentiment.

Here are the steps for Sentiment model building.

- 1)Get the sample form the training data set.
- 2)Sentiment Model Building using below Feature matrix

- Number of words
- Number of characters

- Average word length
- Number of stopwords
- Number of special characters
- Number of numerics
- Number of uppercase words
- Sentiment
- Emojis type (if the customer added comment and emoji too)

4.6 Predict Sales using Sentiment and Semantic (Sales model building and training)

Sales Model Building using below Feature matrix.

- AngryFacecount
- HAHAcount
- LoveCount
- Average sentiment
- ActualSales

4.7 Test and evaluation

- **Accuracy Check the textblob library**

Check the textblob library accuracy which used for semantic analyzed. The subset was getting from training data set. The sentiment was added manually for each and every comment. The purpose was doing this to check the library accuracy. Otherwise, the sentiment model will give the low accuracy.

- **The accuracy of Sentiment model**

After the sentiment model building applied that model to the testing data set and generate the sentiment. The testing data set has sentiment generated by textblob library. Accuracy generated using predicting and real sentiments.

- **The accuracy of Sales model**

After the sales model building, applied that model to the testing data set and generate the predicts sales values. the testing data set has actual sales values

for items which gave by the clothing shop. Accuracy generated using predicting and real sales.

4.8 Identify the correlated item

Each item belongs to the item category. They are Long Sleeved, High Neck, Off shoulder, Jeans Wide Leg, Leggings and Jeans Tight leg. These are the exiting main item categories for that clothing shop. Their main item is clothing but there are selling shoes as well. The shop publishes only new clothing items monthly. So Cloths were only considered for the analyst. Correlated items categories were identified using previous sales details. If the upcoming sales item has any correlation between another item category. Then the sales of other item category are predicted using correlation and sales model.

4.9 Live Dashboard (Correlated item, sales prediction and insight about items) and mobile application

Live Dashboard contains lot of insight about each and every item. It shows correlated item, sales prediction and social media insight for deferent dimension. This is a real time Dashboard since it developed by power BI direct query. Mobile layout is also available for the management.

below is the visualization in the dashboard.

- Monthly item wise positive and negative feedback count.
- Monthly item wise positive and negative sentiments.
- Monthly item sales.
- Monthly item wise predicted sales.
- Comparison between actual sales and predicted sales.
- Comparison between item category wise actual sales.
- Comparison between item category wise actual sales and predicted sales.

Implementation

5.1 Data Collection

- Initially, The extraction code was written by Graph API (<https://www.facebook.com/Thefashionworks/>)
- The row data was extracted from the Facebook page using power and extraction code.

Figure 5.1 shows some extracted raw data from the fan page

- After then staging data was loaded to SQL Server Express edition.

id	pic	comments	emoji	name	created_time	post_id
899862543281910_84550718864957		Wow girl!!!	❤️	Pretty/Beautiful	2018-07-12T18:06:15+000	84550718864957_84724388178525
899862543281910_847228972125467		No good for me		Pretty/Beautiful	2018-07-12T18:54:59+000	847228972125467_847234895497988
899862543281910_84722178781811		Not my type	😂😂😂	Pretty/Beautiful	2018-07-12T18:18:37+000	84722178781811_847237832124401
899862543281910_84722179545703		This is not good		Pretty/Beautiful	2018-07-12T18:09:44+000	84722179545703_847241871212997
899862543281910_847221848781966		Nice one		Pretty/Beautiful	2018-07-12T18:54:39+000	847221848781966_84723482450713
899862543281910_847221848781966		Wowwwww		Pretty/Beautiful	2018-07-12T18:08:02+000	847221848781966_84724405457373
899862543281910_8472179545098		This is funny		Pretty/Beautiful	2018-07-12T18:56:52+000	8472179545098_847237352124435
899862543281910_84722863549114			❤️	Pretty/Beautiful		

Figure 5.1 Raw data from 'Thefashionworks' facebook page

5.2 Data Preprocessing

For any analyze, first of all we need to cleaning the data to extract better features. Achieved this by doing some of the main pre-processing steps using training data. Data preprocessing was done by Python and sql server integration services.

5.2.1 Data Cleaning (Eg. Filter the English comments and reviews)

SSIS used to do data cleaning. Data Flow task was used since it contains the data flow engine that loads data between sources and destinations and provides the functionality for cleaning, summarize, transforming and varying data as it is moved.

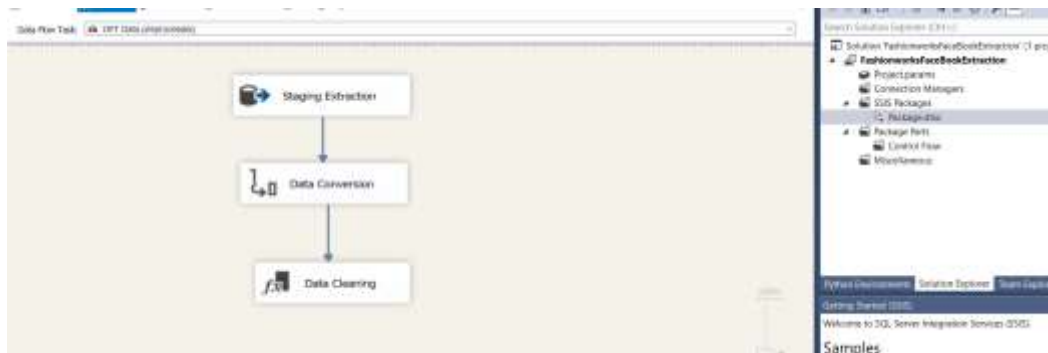


Figure 5.2. Data cleaning

5.2.2 Lower casing

As the first pre-processing step which transform face book data into lower case. This avoided having different copies of the same words. Such as, while computing the word count, 'Good' and 'good' will be taken as distinct words. Figure 5.3 shows the python script for the lower casing.

```
#Convert to lowercase
train['comments'] = train['comments'].apply(lambda x: " ".join(x.lower() for x in x.sp
train['comments'].head()
```



```
Data Modeling Interactive
Environment: Data Modeling  Module: __main__
>>> train['comments'] = train['comments'].apply(lambda x: " ".join(x.lower() for x in x.sp
>>> train['comments'].head()
0    don't you have other sizes
1                hw much
2    don't you have other sizes
3                hw much
4    don't you have other sizes
Name: comments, dtype: object
```

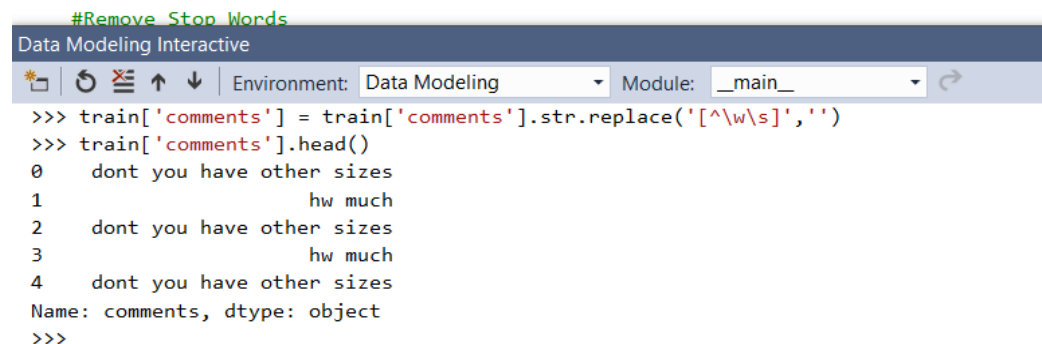
Figure 5.3 Lower casing

5.2.3 Punctuation removal

The next phase was to remove punctuation, as it did not add additional information while processing text data. Then removed all instances since it reduced the size of the training data. Figure 5.3 shows the python script for the remove punctuation.

```
#Remove Punctuation
train['comments'] = train['comments'].str.replace('[^\w\s]', '')
train['comments'].head()
```

```
#Remove Stop Words
```



```
Data Modeling Interactive
Environment: Data Modeling  Module: __main__
>>> train['comments'] = train['comments'].str.replace('[^\w\s]', '')
>>> train['comments'].head()
0    dont you have other sizes
1                hw much
2    dont you have other sizes
3                hw much
4    dont you have other sizes
Name: comments, dtype: object
>>>
```

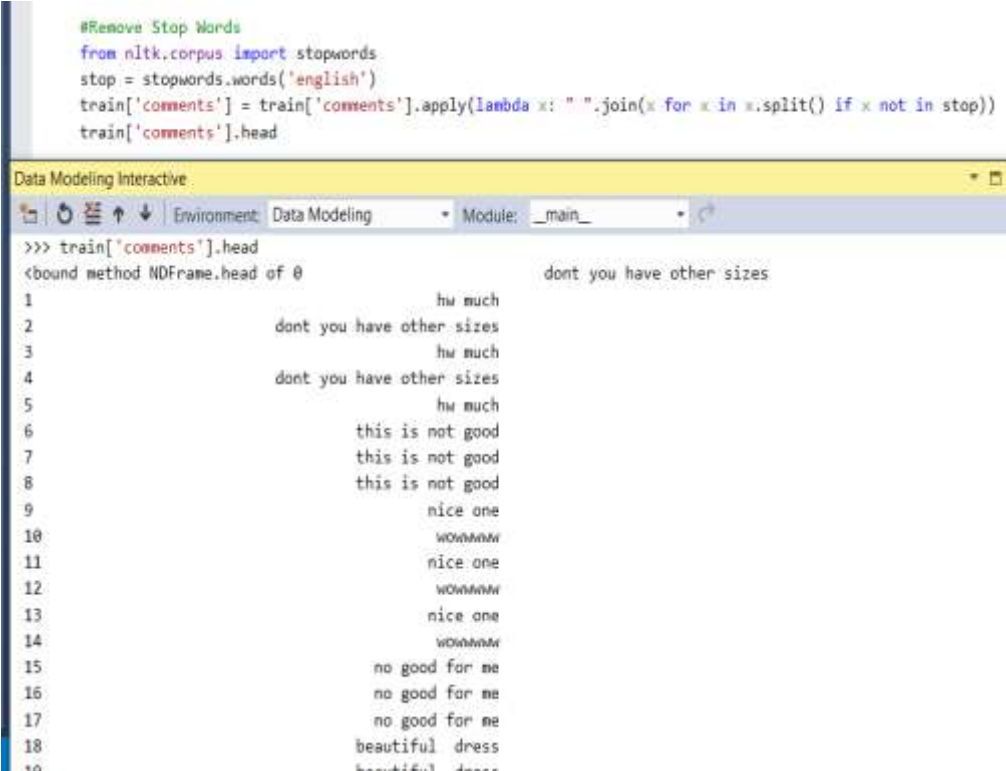
Figure 5.3 Remove Punctuation

As output, all the punctuation has been removed from the training data (Such as '@' and '#')

5.2.3 Stopwords removal

Commonly occurring words (or Stop words) should be eliminated from the text data. For this idea, developer can either create a list of common occurring words themselves or else developer can import predefined libraries. Figure 5.4 shows the python script for the remove stopwords.

```
#Remove Stop Words
from nltk.corpus import stopwords
stop = stopwords.words('english')
train['comments'] = train['comments'].apply(lambda x: " ".join(x for x in x.split() if x not in stop))
train['comments'].head
```

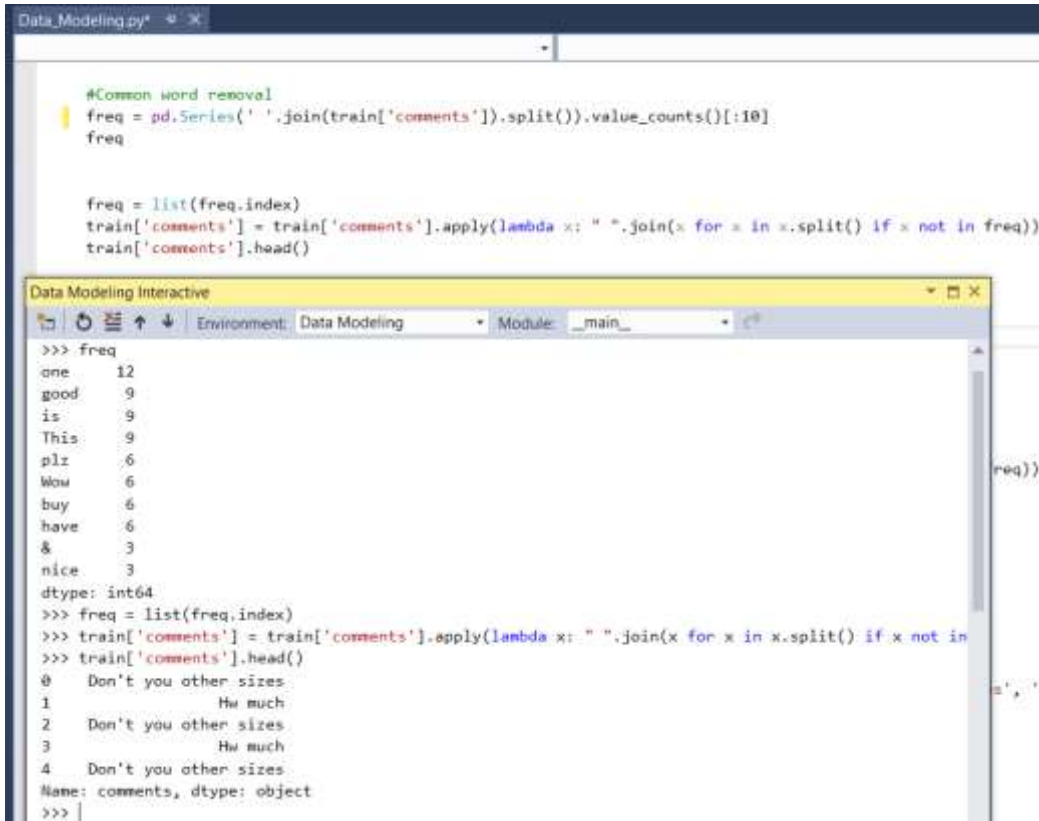


```
>>> train['comments'].head
<bound method NDFrame.head of 0      dont you have other sizes
1      hw much
2      dont you have other sizes
3      hw much
4      dont you have other sizes
5      hw much
6      this is not good
7      this is not good
8      this is not good
9      nice one
10     wowwww
11     nice one
12     wowwww
13     nice one
14     wowwww
15     no good for me
16     no good for me
17     no good for me
18     beautiful dress
19     beautiful dress
```

Figure 5.4 Remove Stopwords

5.2.4 Frequent words removal

Removing commonly occurring words is done for a general sense. Some of the most frequently occurring words (10 words) were taken from my text data then took a call to eliminate or keep. Figure 5.5 shows the python script for the remove common word.



```
Data_Modeling.py *
#Common word removal
freq = pd.Series(' '.join(train['comments']).split()).value_counts()[:10]
freq

freq = list(freq.index)
train['comments'] = train['comments'].apply(lambda x: " ".join(x for x in x.split() if x not in freq))
train['comments'].head()

Data Modeling Interactive
Environment: Data Modeling  Module: __main__  C#
>>> freq
one      12
good     9
is       9
This     9
plz      6
Wow      6
buy      6
have     6
&        3
nice     3
dtype: int64
>>> freq = list(freq.index)
>>> train['comments'] = train['comments'].apply(lambda x: " ".join(x for x in x.split() if x not in
>>> train['comments'].head()
0    Don't you other sizes
1         Hw much
2    Don't you other sizes
3         Hw much
4    Don't you other sizes
Name: comments, dtype: object
>>> |
```

Figure 5.5 Remove Common Word

5.2.5 Rare words removal

According to, eliminated the most occurring words, this part eliminates uncommonly occurring words from the text data. Since they're so uncommon, the relationship between them and other words is affected by noise. Substitution of uncommon words with a more common form and then this provide higher counts. Figure 5.6 shows the python script for the remove rare word.

```

#Rare words removal
freq = pd.Series(' '.join(train['comments']).split()).value_counts()[-10:]
freq

freq = list(freq.index)
train['comments'] = train['comments'].apply(lambda x: " ".join(x for x in x.split() if x not in freq))
train['comments'].head()

```

Figure 5.6 Remove Rare Words

5.2.6 Spelling correction

There were comments and review with a plethora of spelling mistakes.

In that regard, spelling correction is very important for data pre-processing step since this also will reducing different copies of words. Such as, “Beautiful” and “Beutiful” will be taken as distinct words even if they are commented in the same sense. To fulfill this, textblob library was used. Figure 5.7 shows the python script for the spelling correction.

```

#Spelling Correction
from textblob import TextBlob
train['comments'][:5].apply(lambda x: str(TextBlob(x).correct()))

```

Figure 5.7 Spelling Correction

Some of words were time to time used in their briefed (abbreviated) form. For some instance, ‘your’ is commented like ‘ur’. Using rule base condition, those word were converted to meaningful words. It should have to treated this before the spelling correction process, or else these words might be converted into other word.

5.2.7 Tokenization

Tokenization uses to breaking the text into a sequence of words or sentences. In my case, *textblob* library was referred to convert my Facebook raw data into a blob and then transformed them into a series of words. Figure 5.8 shows the python script for tokenization.

```
#Tokenization
nltk.data.path.append(r"C:\nltk_data")
import nltk
nltk.download('punkt')
TextBlob(train['comments'][1]).words
```

Figure 5.8 Tokenization

5.2.8 Stemming

Stemming used to the eliminate of suffices, such as “s”, “ly”, “ing”, etc. by a simple rule-based approach. “PorterStemmer” library was used from the NLTK library. Figure 5.9 shows the python script for stemming.



```
#Stemming
from nltk.stem import PorterStemmer
st = PorterStemmer()
train['comments'][:5].apply(lambda x: " ".join([st.stem(word) for word in x.split()]))
```

Data Modeling Interactive

```
>>> from nltk.stem import PorterStemmer
>>> st = PorterStemmer()
>>> train['comments'][:5].apply(lambda x: " ".join([st.stem(word) for word in x.split()]))
0 don't size
```

Figure 5.9 Stemming

5.2.9 Lemmatization

Lemmatization is a more performing option than stemming since it transforms the word into its root word, rather than just removing the suffices. It produces use of the vocabulary and does a morphological analysis to obtain the root

word [8]. ‘Wordnet’ predefine library used to lemmatization. Because of this, lemmatization was used over stemming. Figure 5.10 shows the python script for lemmatization.

```
#Lemmatization
from textblob import Word
nltk.data.path.append(r"C:\nltk_data")
import nltk
nltk.download('wordnet')

train['comments'] = train['comments'].apply(lambda x: " ".join([Word(word).lemmatize() for word in x.split()]))
train['comments'].head()
```

Data Modellog Interactive

Environment: Data Modeling - Module: _main_ -

```
>>> train['comments'] = train['comments'].apply(lambda x: " ".join([Word(word).lemmatize() for word in x.split()]))
>>> train['comments'].head()
0    Don't size
```

Figure 5.10 Lemmatization

5.3 Feature Extraction using Text Data

Preprocessed Data is used as input for this part. In this section, showed distinct feature extraction methods, beginning with some basic techniques which guide into complex Natural Language Processing techniques.

Below Step followed for feature extraction using text data

5.3.1 Number of words

One of the most fundamental features that extracted was the number of words in each comments. The basic intuition behind this is that generally, the negative sentiments contain a lesser amount of words than the positive ones. [8].

To do this, apparently split function in python was used. Figure 5.11 shows the python script for generate number of word in the comments.

```

>>> train['word_count'] = train['comments'].apply(lambda x: len(str(x).split(" ")))
>>> train[['comments', 'word_count']].head()
      comments  word_count
0  Don't you have other sizes      5
1                How much      2
2  Don't you have other sizes      5
3                How about the price  4
4  Don't you have other colours      5

```

Figure 5.11 Number of words in the comments

5.3.2 Number of characters

This feature is one of fundamental feature like previous feature. Number of characters in every comments were calculated. This is done by counting the length of the comments. The space calculation can be removed as required. Figure 5.12 shows the python script for generate number of characters in the comments.

```

>>> train['char_count'] = train['comments'].str.len() ## this also includes spaces
>>> train[['comments', 'char_count']].head()
      comments  char_count
0  Don't you have other sizes      26
1                How much      8
2  Don't you have other sizes      26
3                How about the price  19
4  Don't you have other colours      28

```

Figure 5.12 Number of characters in the comments

5.3.3 Average word length

Calculating the average word length of each comment was the another valuable feature. This helped to improve model. Sum of the length of all the words has taken and divided it by the total length of the comments. Figure 5.13 shows the python script for generate average word length in the comments.

```

>>> train['avg_word'] = train['comments'].apply(lambda x: avg_word(x))
>>> train[['comments', 'avg_word']].head()
      comments  avg_word
0  Don't you have other sizes    4.4
1                How much      3.5
2  Don't you have other sizes    4.4
3                How about the price  4.0
4  Don't you have other colours    4.8

```

Figure 5.13 Average word length in the comments

5.3.4 Number of stopwords

Generally, while solving an NLP problem, the first thing we do is to remove the stopwords. But sometimes calculating the number of stopwords can also give some extra information which we might have been losing before. [8]

Stopwords library was referred from NLTK, which is a primary NLP library in python. Figure 5.14 shows the python script for generate number of stopword.

```

#Remove Stop Words
from nltk.corpus import stopwords
stop = stopwords.words('english')
train['comments'] = train['comments'].apply(lambda x: " ".join([w for w in x.split() if w not in stop]))
train['comments'].head

```

Figure 5.14 count number of stop word from comments

5.3.5 Number of special characters

One more useful feature which that can be retrieve from a comments is counting the number of hashtags or mentions present in it. This feature used to retrieve more details from my text data.

Here, 'starts with' function was used since hashtags (or mentions) always manifest in the beginning of a word. Figure 5.15 shows the python script for generate number of special character word from comments.

```

#Number of Special characters
train['hashtags'] = train['comments'].apply(lambda x: len([w for w in x.split() if w.startswith('#')]))
train[['comments', 'hashtags']].head

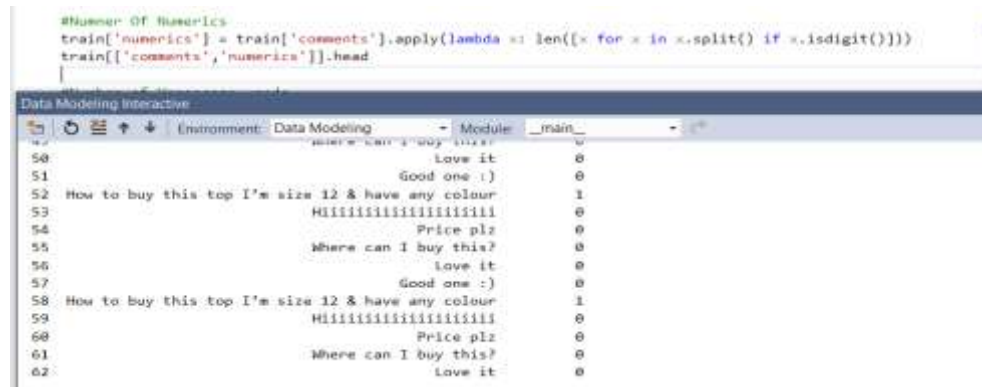
```

Figure 5.15 Number of special character word from comments

5.3.6 Number of numeric

Just like word's numbers were counted, the number of numeric which are present in the comments were calculated. It was useful feature for the model. Figure 5.16 shows the python script for generate number of numeric from comments

```
#Number of Numerics
train['numerics'] = train['comments'].apply(lambda x: len([x for x in x.split() if x.isdigit()]))
train[['comments', 'numerics']].head()
```



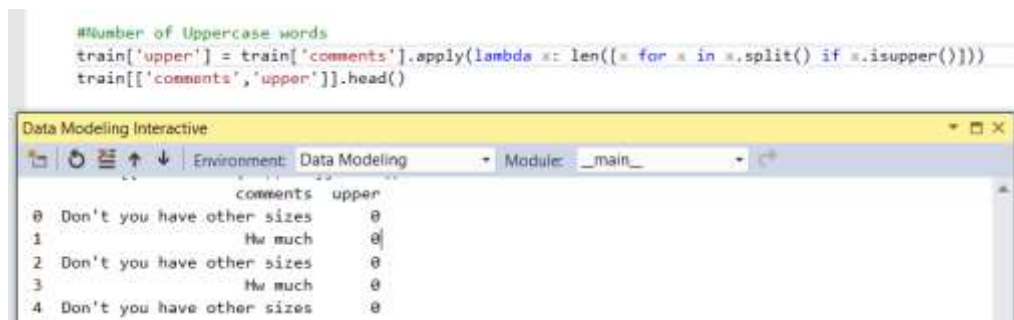
Index	comments	numerics
50	Love it	0
51	Good one :)	0
52	How to buy this top I'm size 12 & have any colour	1
53	Hiiiiiiiiiiiiiiiiiiii	0
54	Price plz	0
55	Where can I buy this?	0
56	Love it	0
57	Good one :)	0
58	How to buy this top I'm size 12 & have any colour	1
59	Hiiiiiiiiiiiiiiiiiiii	0
60	Price plz	0
61	Where can I buy this?	0
62	Love it	0

Figure 5.16 generate number of numeric from comments

5.3.7 Number of uppercase words

Anger or rage is quite often expressed by writing in UPPERCASE words which makes this a necessary operation to identify those words. [8]. Figure 5.17 shows generate number of upper case word in the comments.

```
#Number of Uppercase words
train['upper'] = train['comments'].apply(lambda x: len([x for x in x.split() if x.isupper()]))
train[['comments', 'upper']].head()
```



Index	comments	upper
0	Don't you have other sizes	0
1	Hw much	0
2	Don't you have other sizes	0
3	Hw much	0
4	Don't you have other sizes	0

Figure 5.17 Generate number of upper case word in the comments.

5.4 Split training and testing data

After the feature matrix creation next step is split training and testing data set.

These are libraries which were imported:

- Pandas - to get the data file as a Pandas data frame and analyze the data.
- Sklearn - to load the linear model for run a linear regression.
- Sklearn (train_test_split) - to split to training and test sets.
- Matplotlib (pyplot) - to order to plot graphs of the data.

After the loading preprocessed data set, it turned into a data frame and commentate the columns' names.

'train_test_split' function was used to make the split. The *test_size=0.2* among the function indicated the percentage of the data that allocate for testing. It's usually around 80/20 or 70/30([9]). 80/20 was used since the data set was not much huge and retaining the big data set for modeling is better. Figure 5.18 shows the python code for splitting the training and testing data set.

```
#Load the Dataset
columns = "char_count word_count avg_word stopwords hastags numerics upper LOVE ANGRY HAHA".split()#Decalre the colum names

dataset =train[['char_count','word_count','avg_word','stopwords','hastags','numerics','upper','LOVE','ANGRY','HAHA']]#Load Data set
dataset

#Load dataset as pandas Dataframe
df=pd.DataFrame(dataset,columns=columns)
y=train[['sentiment']]#define dependant variable
X_train,X_test,y_train,y_test=train_test_split(df,y,test_size=0.2)#create training and testing dataset
```

Figure 5.18 splitting the training and testing data set.

5.5 Sentiment Analysis (Sentiment model building and training)

5.5.1 Sentiment Analysis using textblob library

Separate feature detecting the sentiment was done using the textblob library. Below, returns were subjectivity and tuple representing polarity of each comments. Polarity as it indicated the sentiment as value nearer to one(1) means a positive sentiment and values nearer to minus one (-1) means a negative sentiment. Figure 5.19 shows the python code for detecting the sentiment using the textblob library.

```
#Sentiment Analysis
import textblob
train['comments'][:63].apply(lambda x: TextBlob(x).sentiment)

train['sentiment'] = train['comments'].apply(lambda x: TextBlob(x).sentiment[0] )
train[['comments','sentiment']].head()
```

Data Modeling Interactive

```
>>> train['sentiment'] = train['comments'].apply(lambda x: TextBlob(x).sentiment[0] )
>>> train[['comments','sentiment']]
```

	comments	sentiment
0	Don't you have other sizes	-0.125
1	How much	0.200
2	Don't you have other sizes	-0.125
3	How about the price	0.000

Figure 5.19 detecting the sentiment using the textblob library.

Predicted sentiment also worked as a dependent variable for building a sentiment model. Data set and dependent variable data divided into training and testing data. Figure 5.20 shows the python code for splitting and setting the dependent variable.

```
#Load the Dataset
columns = "char_count word_count avg_word stopwords hastags numerics upper LOVE ANGRY HAHA".split()#Decalre the colum names

dataset=train[['char_count','word_count','avg_word','stopwords','hastags','numerics','upper','LOVE','ANGRY','HAHA']]#Load Data set
dataset

#Load dataset as pandas Dataframe
df=pd.DataFrame(dataset,columns=columns)
y=train[['sentiment']]#define dependant variable
X_train,X_test,y_train,y_test=train_test_split(df,y,test_size=0.2)#create training and testing dataset
```

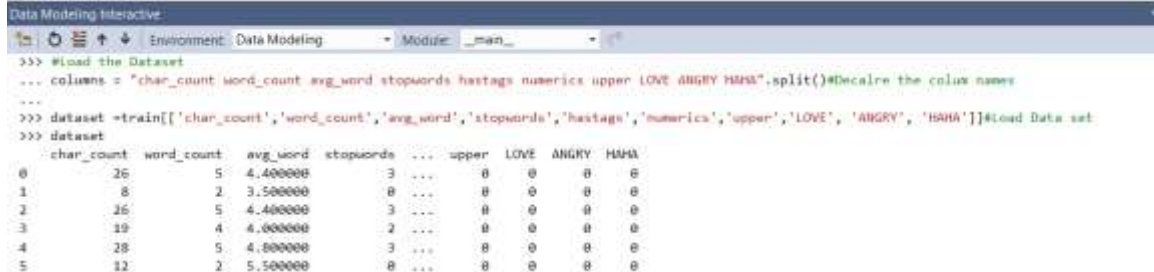
Figure 5.20 python code for splitting and setting the dependent variable

The generated sentiment was set as the dependent variable. All other variables (chat count, word count, stopwords, hashtag, numeric and upper case word) set as feature variable Figure 5.21 shows the python code for feature matrix creation.

```
import pandas as pd
from sklearn import datasets, linear_model
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt

#Load the Dataset
columns = "char_count word_count avg_word stopwords hashtags numerics upper LOVE ANGRY Haha".split()#Decalre the colux names

dataset =train[['char_count', 'word_count', 'avg_word', 'stopwords', 'hashtags', 'numerics', 'upper', 'LOVE', 'ANGRY', 'Haha']]#Load Data set
dataset
```



	char_count	word_count	avg_word	stopwords	...	upper	LOVE	ANGRY	Haha
0	26	5	4.400000	3	...	0	0	0	0
1	8	2	3.500000	0	...	0	0	0	0
2	26	5	4.400000	3	...	0	0	0	0
3	19	4	4.000000	2	...	0	0	0	0
4	28	5	4.800000	3	...	0	0	0	0
5	12	2	5.500000	0	...	0	0	0	0

Figure 5.21 the python code for feature matrix creation.

Then The model was fit based on the training data. Figure 5.22 shows the python code fitting and training model.

```
#Load dataset as pandas Dataframe
df=pd.DataFrame(dataset,columns=columns)
y=train[['sentiment']]#define dependant variable
X_train,X_test,y_train,y_test=train_test_split(df,y,test_size=0.2)#create training and testing dataset

X_train.shape
X_test.shape
y_train.shape
y_test.shape

#fit a model
lm=linear_model.LinearRegression()

model =lm.fit(X_train,y_train)
```

Figure 5.22 python code fitting and training model

Model was fit to the training data. After that prediction was created to test data. Below shows some of the predictions are (Figure 5.23)

```
>>> prediction[0:5]
array([0.34501108, 0.33308407, 0.17047741, 0.08019986, 0.14776025])
```

Figure 5.23 Predict the test data using model

Then the model was plotted. Figure 5.24 shows the python script for model plotting and then the accuracy.

```
>>> plt.scatter(y_test,prediction)
<matplotlib.collections.PathCollection object at 0x000000812E61B710>
>>> plt.xlabel("True Values")
Text(0.5, 0, 'True Values')
>>> plt.ylabel("prediction")
Text(0, 0.5, 'prediction')
>>> plt.show()
>>> model.score(X_test,y_test)
0.6666413522774699
```

Figure 5.24 plotting the model

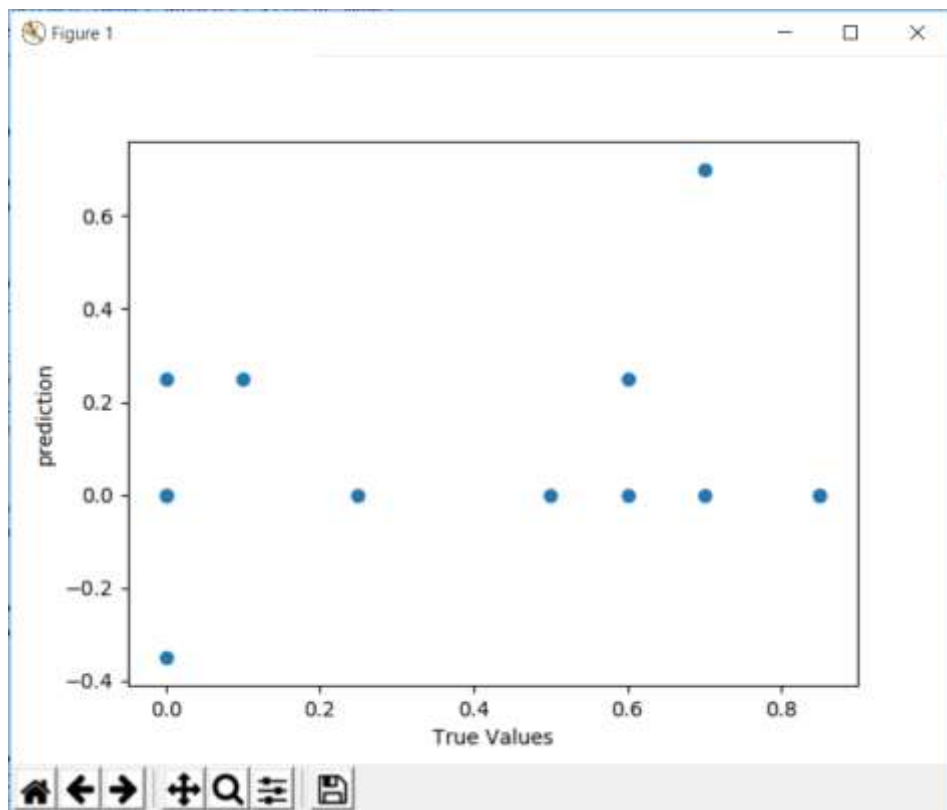
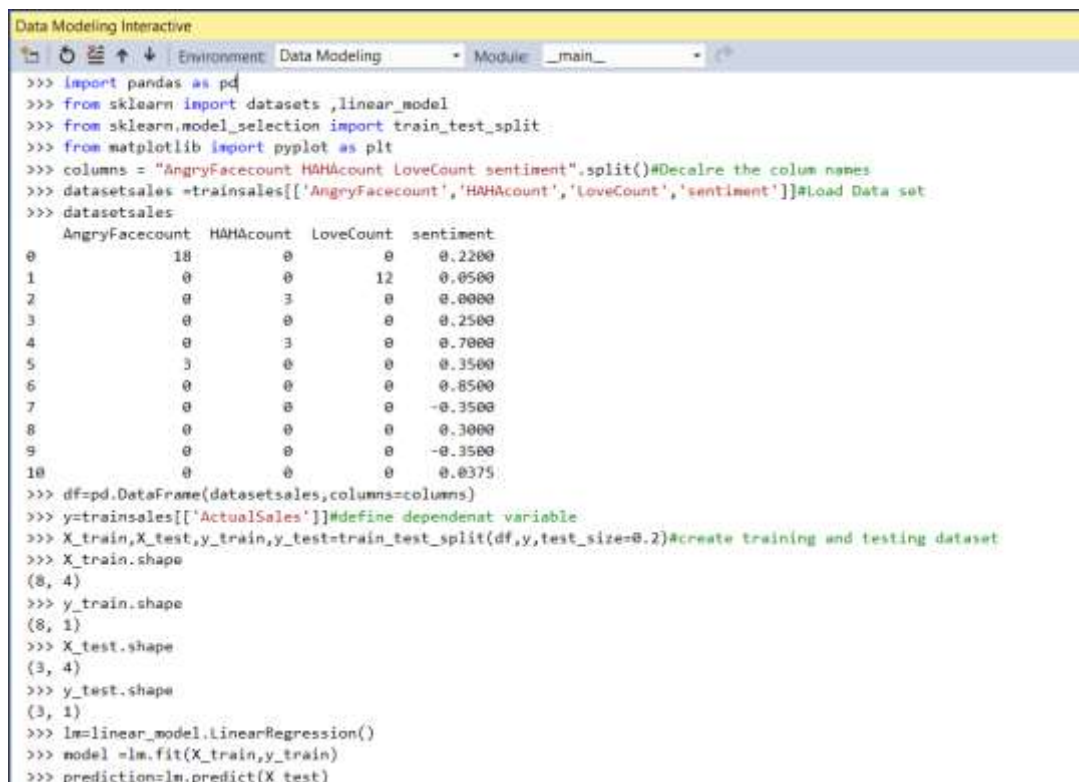


Figure 5.25 plot sentiment model

5.6 Predict Sales using Sentiment and Semantic (Sales model building and training)

Actual sales were set as the dependent variable. All other variables (Angry count, HAHA count ,love count and sentiment) set as independent variable .average sentiment was created for each, and every item using sentiment model .Figure 5.26 shows the python scripts for dependent/independent variable setting and fitting the model.



```
Data Modeling Interactive
>>> import pandas as pd
>>> from sklearn import datasets ,linear_model
>>> from sklearn.model_selection import train_test_split
>>> from matplotlib import pyplot as plt
>>> columns = "AngryFacecount HAHAcount LoveCount sentiment".split()#Decalre the colum names
>>> datasetsales =trainsales[['AngryFacecount','HAHAcount','LoveCount','sentiment']]#Load Data set
>>> datasetsales
   AngryFacecount  HAHAcount  LoveCount  sentiment
0                18         0          0     0.2200
1                 0         0         12     0.0500
2                 0         3          0     0.0000
3                 0         0          0     0.2500
4                 0         3          0     0.7000
5                 3         0          0     0.3500
6                 0         0          0     0.8500
7                 0         0          0    -0.3500
8                 0         0          0     0.3000
9                 0         0          0    -0.3500
10                0         0          0     0.0375
>>> df=pd.DataFrame(datasetsales,columns=columns)
>>> y=trainsales[['ActualSales']]#define dependant variable
>>> X_train,X_test,y_train,y_test=train_test_split(df,y,test_size=0.2)#create training and testing dataset
>>> X_train.shape
(8, 4)
>>> y_train.shape
(8, 1)
>>> X_test.shape
(3, 4)
>>> y_test.shape
(3, 1)
>>> lm=linear_model.LinearRegression()
>>> model =lm.fit(X_train,y_train)
>>> prediction=lm.predict(X_test)
```

Figure 5.26 Python scripts for dependent/independent variable setting and fitting the model

The model was fit based on the training data and tried to predict the test data. Below shows some of the predictions. (Figure 5.27)

```
>>> prediction[0:5]
array([[5198.50433639],
       [3913.86001754],
       [3579.85249464]])
>>> plt.scatter(y_test,prediction)
<matplotlib.collections.PathCollection object at 0x00000009995F64A8>
>>> plt.xlabel("True Values")
Text(0.5, 0, 'True Values')
>>> plt.ylabel("prediction")
Text(0, 0.5, 'prediction')
>>> plt.show()
>>> model.score(X_test,y_test)
0.69191821114448955
>>>
```

Figure 5.27 some of sales prediction for testing data set

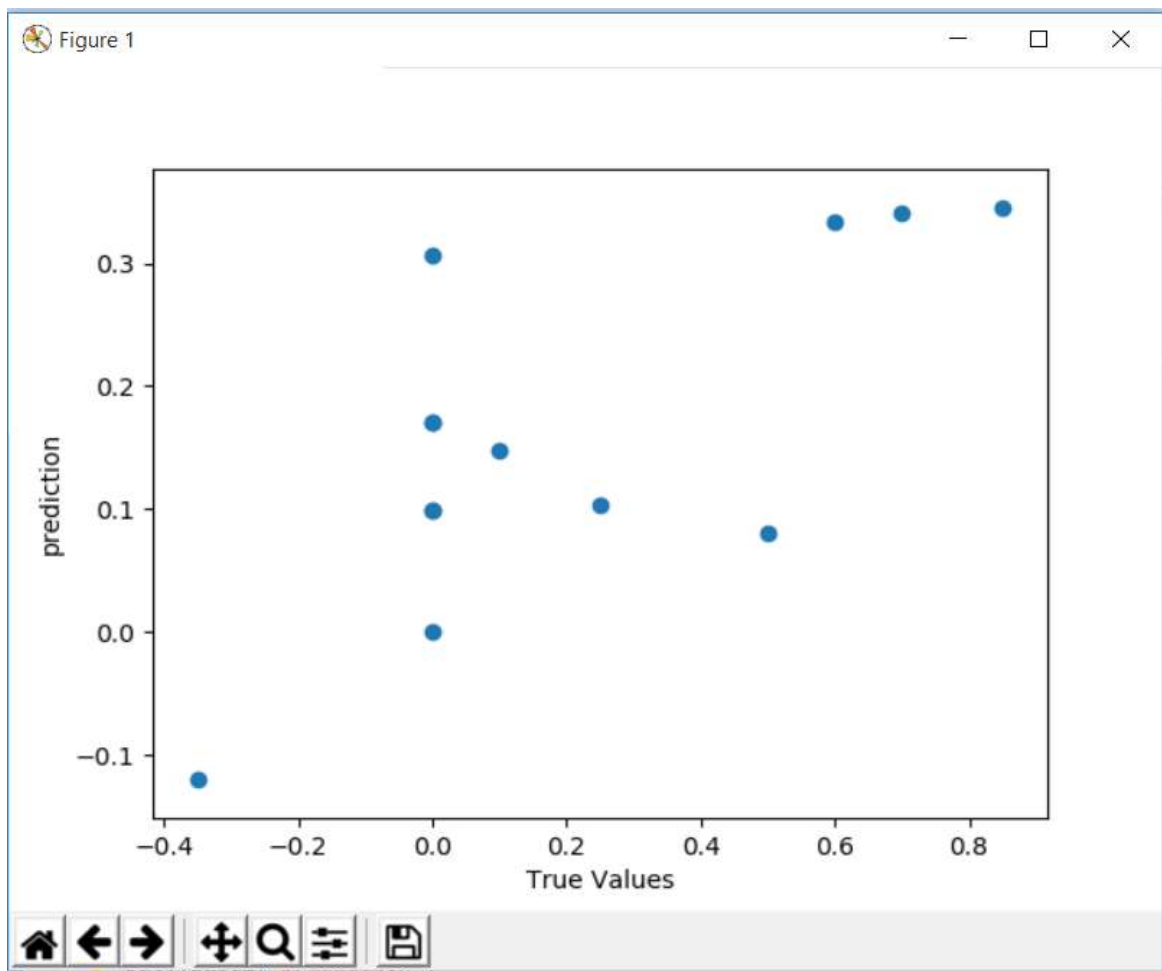
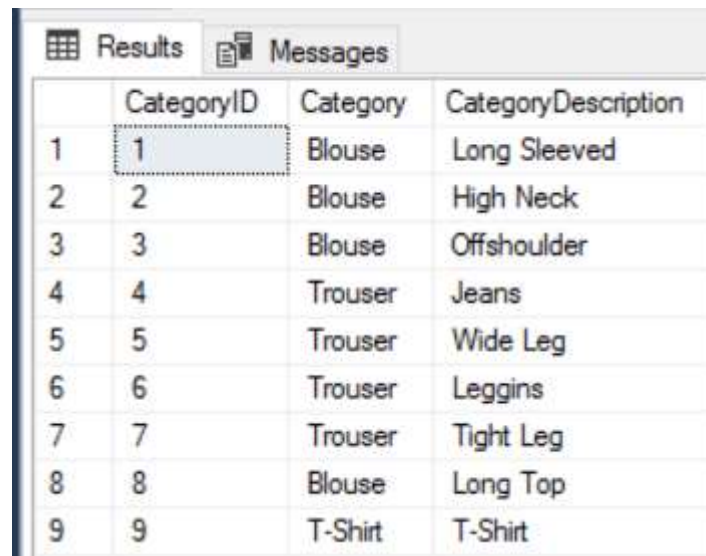


Figure 5.28 plotting the sales model

5.7 Predicting Sales item and correlated item

Each item belongs to item category. They are Long Sleeved, High Neck, Off shoulder, Jeans, Wide Leg, Leggings, Tight Leg, Long Top and T-shirt. These are the exiting main item categories for that clothing shop. Their main item is clothing but there are selling shoes as well. Figure 5.29 shows exiting item category



The screenshot shows a window with two tabs: 'Results' and 'Messages'. The 'Results' tab is active and displays a table with four columns: an index, 'CategoryID', 'Category', and 'CategoryDescription'. The table contains nine rows of data.

	CategoryID	Category	CategoryDescription
1	1	Blouse	Long Sleeved
2	2	Blouse	High Neck
3	3	Blouse	Offshoulder
4	4	Trouser	Jeans
5	5	Trouser	Wide Leg
6	6	Trouser	Leggins
7	7	Trouser	Tight Leg
8	8	Blouse	Long Top
9	9	T-Shirt	T-Shirt

Figure 5.29 exiting item category

The shop publishes only new clothing items monthly. Correlated items categories were identified using previous sales details. If the upcoming sales item has any correlation between another item category. Then the sales of other item category are predicted using correlation and sales model.

‘`scipy.stats.pearsonr(x, y)`’ library was used for check correlation.

The Pearson correlation coefficient measures the linear relationship between the two datasets. Strictly speaking, Pearson’s correlation requires that each dataset is normally distributed. Like other correlation coefficients, this one varies between -1 and +1 with 0 implying no correlation. Correlations of -1 or +1 imply an exact linear relationship. Positive correlations imply that as x increases, so does y. Negative correlations imply that as x increases, y decreases. [10]

When we check through the previous month's sales, a positive correlation between some item category was identified. Figure 5.30 shows the python script and results.

```
0.1/1/01/01000000
>>> import scipy
>>> from scipy.stats import pearsonr
>>> x = scipy.array([3034 ,3019, 3019, 349])
>>> y = scipy.array([2014, 3019 ,405 ,324])
>>> r_row, p_value = pearsonr(x, y)
>>> r_row
0.5692523958264609
>>> import scipy
>>> from scipy.stats import pearsonr
>>> x = scipy.array([3034 ,3019, 3019, 349])
>>> y = scipy.array([1092, 2092 ,691, 290])
>>> r_row, p_value = pearsonr(x, y)
>>> r_row
0.6467326894354755
>>>
```

Figure 5.30 check correlation - python script and results.

Using the sales model, the predicted sale count is generated for the correlated item as well.

5.8 Live Dashboard (Correlated item, sales prediction and insight about items) and mobile application

Live Dashboard contains a lot of insight about each and every item. It shows correlated item, sales prediction and social media insight for deferent dimension. This is a real-time Dashboard since it developed by power BI direct query. Mobile layout is also available for the management.

below is the visualization in the dashboard.

- Monthly item wise positive and negative feedback count.
- Monthly item wise positive and negative sentiments.
- Monthly item sales.
- Monthly item wise predicted sales.
- Comparison between actual sales and predicted sales.

- Comparison between item category wise actual sales.
- Comparison between item category wise actual sales and predicted sales.

Below figure 5.31 shows some charts of dashboards.

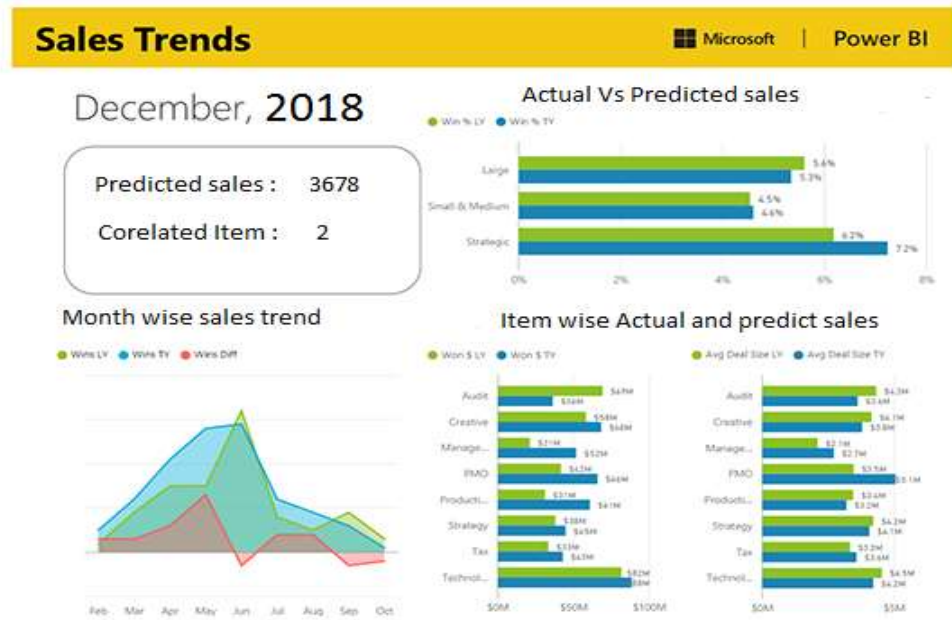


Figure 5.31 Some charts of dashboard

Evaluation

6.1 Accuracy Check the textblob library

The first evaluation was checking the textblob library accuracy which used for semantic analyzed. The subset was getting from training data set. The sentiment was added manually for each and every comments. The purpose was doing this is to check the library accuracy. Otherwise the sentiment model will give the low accuracy.

6.2 Accuracy of Sentiment model

After the sentiment model building applied that model to the testing data set and generate the sentiment. The testing data set has sentiment generated by textblob library. Accuracy generated using predicting (sentiment model) and actual sentiments (generated by textblob). Figure 6.1 shows the python scrip for generate accuracy of sentiment model.

```
>>> plt.scatter(y_test,prediction)
<matplotlib.collections.PathCollection object at 0x000000812E61B710>
>>> plt.xlabel("True Values")
Text(0.5, 0, 'True Values')
>>> plt.ylabel("prediction")
Text(0, 0.5, 'prediction')
>>> plt.show()
>>> model.score(X_test,y_test)
0.6666413522774699
```

Figure 6.1 Checking Accuracy

The accuracy was 66% for the sentiment model. Model was trained using four-month data. If there is any bigger data set, the accuracy will be improved.

6.3 Accuracy of Sales model

After the sales model building applied that model to the testing data set and generate the predicted sales values. The testing data set has actual sales values for items which gave by the clothing shop. Accuracy generated using predicting and real sales. 6.2 shows the python scrip for generate accuracy of sales model.

```
>>> prediction[0:5]
array([[5198.50433639],
       [3913.86001754],
       [3579.85249464]])
>>> plt.scatter(y_test,prediction)
<matplotlib.collections.PathCollection object at 0x00000009995F64A8>
>>> plt.xlabel("True Values")
Text(0.5, 0, 'True Values')
>>> plt.ylabel("prediction")
Text(0, 0.5, 'prediction')
>>> plt.show()
>>> model.score(X_test,y_test)
0.69191821114448955
>>>
```

Figure 6.2 Shows the python scrip for generate accuracy of sales model

The accuracy was 69% for the sales model. Model was trained using four-month data.

Conclusion and further work

Sentiment Analysis can help gain insight into so much about how consumers feel about a brand, about brands campaigns, products and services. There are so many tools that can help IT departments or marketing departments. But as always it depends on the business, time and the most important in the majority of the company's budget and resources. Fashion product review is more subjective than other product.

Because of this, the complexity of fashion product review is higher than other product and difficult to analyze. There are no well develop a mechanism to analyze subjective social media data. There is no 24/7 live dashboard for sentiments from Facebook comment towards the brand.

7.1 Overall Achievement

Throughout the research data was transformed from the Facebook platform to live dashboard. There was two models with nearly 65% accuracy. Management staff had to be more warn and alert about the negative comments and posts. Because that can be harmful to the image of a brand or product. Management staff can be arranging their production line according to predicted sales. That is cost effective way to fashion production. Research help to reduce the market analysis duration and provide higher managers the feedback in order to align their business operation accordingly. Insight can be used to organizations to gain advantages Bring the real-time, can determine if the identification of the feeling is positive, negative and act at the moment.

7.2 Achievement of each objective

- Analyze the comments, emojis and review for upcoming items.
- Predict upcoming sales through qualitative and quantitative social media data and previous months' sales.

- Identify the correlated item that can be a sale with top selling items category.
- It will make a cost-effective solution for the Fashion Industry.
- Real-time dashboard and mobile application for top management.
- Provide a well develop a mechanism to analyze subjective social media data.

7.3 Problem encountered

As discussed with many clothing stores, one of them agreed to give the sales detail and authorization to their Facebook page. It contains the latest four months of sales details. So the data volume is not big, and it got an impact on the model accuracy. Sentiment model accuracy was 66%, and the sales model accuracy was 69%.

7.4 Limitation

For the correlation item category generation, correlated item categories were identified. Furthermore, the analyzer can generate correlated by considering item color, size and item category wise.

7.5 Further work

Thought out the research The only data extraction source was Facebook only. But the fashion clothing shops might have Instagram, Twitter and other social media pages as well. AS further work data can be extracted form difference social media network and integrated those data and get into one common structure. This day most of the people trend to use Instagram. Using Instagram data will be able to extract valuable insight as well.

References

- [1] L. M. Jose, A semantic graph based approach on interest extraction from user generated texts in social media, India, 2017.
- [2] A. Teixeira, "Data extraction and preparation to perform a sentiment analysis using open source tools: The example of a Facebook fashion brand page," Portugal, 2017.
- [3] C. Lee, "Advances in Software Engineering," in International Conference, ASEA 2008, China, 2008.
- [4] J. Beger, "Arousal Increases Social Transmission of Information," 2011.
- [5] M. Stelzner, "Technology and Competitive Advantage," 2014.
- [6] M. Godsay, "The Process of Sentiment Analysis: A Study," International Journal of Computer Applications, 2015.
- [7] <https://dev.to/jjsantos/first-impression-about-facebooks-graph-api--59i5>
- [8] <https://www.analyticsvidhya.com/blog/2018/02/the-different-methods-deal-text-data-predictive-python/>
- [9] <https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6>
- [10] <https://docs.scipy.org/doc/scipy0.14.0/reference/generated/scipy.stats.pearsonr.html>