# Sentiment analysis on top selling Smart Tv reviews

**November 6**

**Authored by: Sriharsha Vajjala (A20398670)**

**Shiva Chandan Reddy Kancharla(A20398806)**

**Santhosh Chittiprolu (A20377479)**

## **TABLE OF CONTENTS**

**Executive summary………………………………………………………………………………….**

**Statement of scope………………………………………………………………………………….**

**Project Schedule………………………………………………………………………………………**

**Data Preparation…………………………………………………………………………………......**

**Data Access……………………………………………………………………………………………..**

**Data Cleaning……………………………………………………………………………………………**

**Data Transformation…………………………………………………………………………………**

**Data Reduction………………………………………………………………………………………….**

**Data Consolidation……………………………………………………………………………………**

**Data Dictionary…………………………………………………………………………………………**

**Text mining and Sentiment analysis………………………………………………………….**

**Classification Model………………………………………………………………………………….**

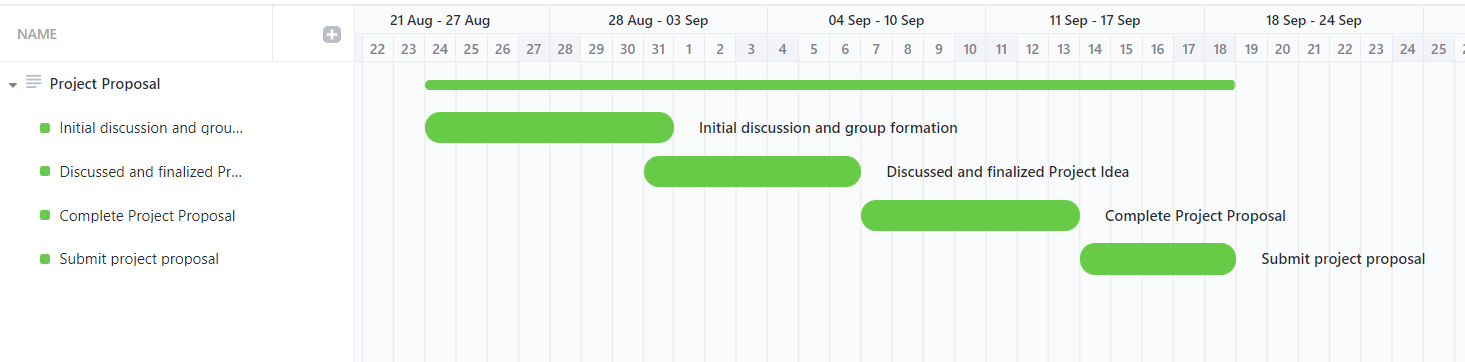
**Conclusion………………………………………………………………………………………………..**

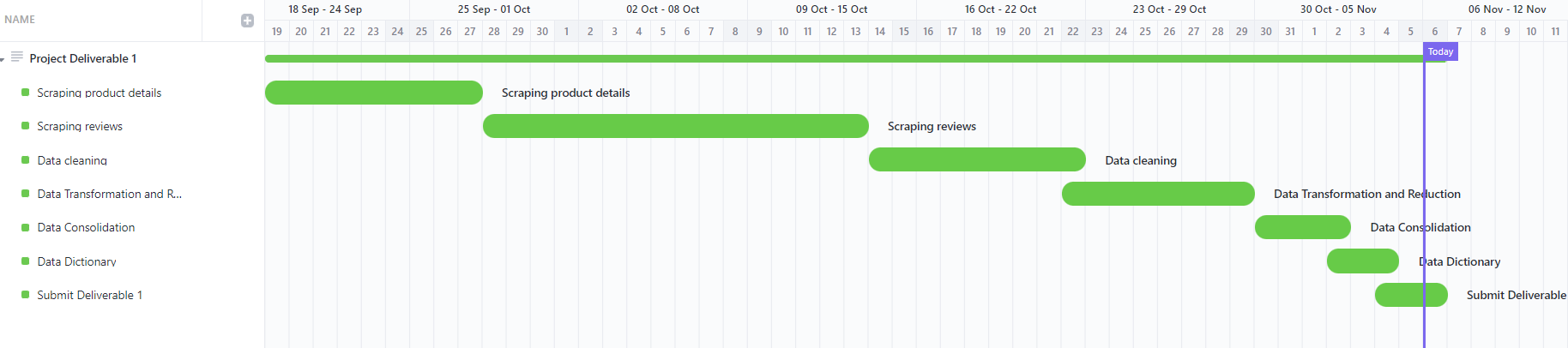
**EXECUTIVE SUMMARY**

Amazon is most used e-commerce website to purchase electronic product. Global Smart Tv market is valued at 187.1 billion dollars in 2021 and it is expected to grow at least 10% over the next 5-10 years. It is very important to understand how customers are feeling about the Smart Tv’s they bought and their expectations. Our project works on examining amazon reviews for the top selling Smart Tv’s in USA. We want to understand opinions of the customers to analyse what kind of features they wants. We use descriptive and statistical analysis on this data to find insights. This proposed issue helps electronic manufacturers to come up with better and quality products that users expect and the features they like. The sentiment analysis on reviews will reveal the most common emotions felt by people and categorise the topics from the data.

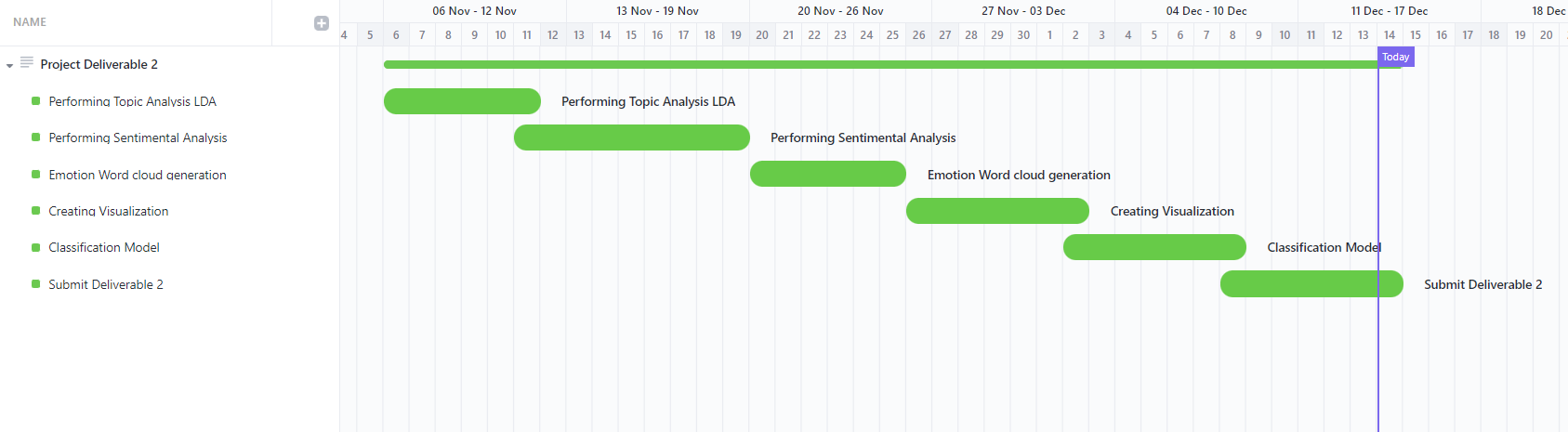
**PROJECT SCHEDULE**

The initial GANTT Chart Schedule, which includes all of the project's activities and tasks for first deliverable, is shown below. We have included the detailed Gantt chart from initial discussion to Submission of proposal. And in deliverable 1 Gantt chart we included all the steps and time spent on each steps like scraping the data, data cleaning, data transformation and reduction to submission.





Gantt chart for Project deliverables 2



**STATEMENT OF SCOPE**

**STATEMENT**

The purpose of this project is to identify the emotions felt by customers that bought the Smart Tv’s. For this project, we will be analysing data which is scraped from the amazon website using the product reviews on Smart Tv’s.

**PROJECT OBJECTIVES**

The project objectives include the following:

* Extracting amazon review data for the top selling Smart Tv’s along with title, size of the tv, brand and review comments.
* Performing descriptive statistical analysis on the data to identify insights on customers reviews and emotions.
* Performing sentiment analysis which reveal the positive and negative reviews posted by customers
* Analysing the review text using text analysis and categorise the common topics and issues raised by customers.

**UNIT OF ANALYSIS**

Our project analyses sentiment from amazon reviews for top selling Smart Tv’s. So, review body (review comments) is our unit of analysis.

**DATA PREPARATION**

We have scraped the data from amazon website for the top selling Smart Tv’s in USA. The mail goal of this project is to identify the most frequent emotions felt by customers that bought the Smart Tv’s from the amazon website.

Final data file which contains product details and reviews:

<https://github.com/msis5193-pds1-2022fall/project-deliverable-2-cowboys/blob/main/data/TV_Reviews.csv> :

Python script to scrape reviews:

<https://github.com/msis5193-pds1-2022fall/project-deliverable-2-cowboys/blob/main/code/Review_details.py> **:**

Python script to scrape product details:

<https://github.com/msis5193-pds1-2022fall/project-deliverable-2-cowboys/blob/main/code/scraper_product_details.py> :

Script to clean data:

<https://github.com/msis5193-pds1-2022fall/project-deliverable-2-cowboys/blob/main/code/Data_cleaning.py>

<https://github.com/msis5193-pds1-2022fall/project-deliverable-2-cowboys/blob/main/code/Data_cleaning_on_review.py>

<https://github.com/msis5193-pds1-2022fall/project-deliverable-2-cowboys/blob/main/code/Data_cleaning_2.py>

**DATA ACCESS**

We have obtained the data from the amazon website. We have found a page for top selling 100 Smart Tv’s on amazon website in the below link

<https://www.amazon.com/Best-Sellers-Televisions/zgbs/electronics/172659>

We have written a code to extract the product details including product ID, title, price, brand and resolution. We used selenium to click on the next page at the bottom of this page to extract the product details from the remaining 50 Smart Tv’s. We installed chrome driver and using selenium it automatically launches a chrome window to extract the product details and uses this product details. We build URL based on the product id, then it navigates to URL to scrape the review.

SampleURL:<https://www.amazon.com/Sony-Inch-Ultra-X80K-KD65X80K/product-reviews/B09P4C85N8/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_review>

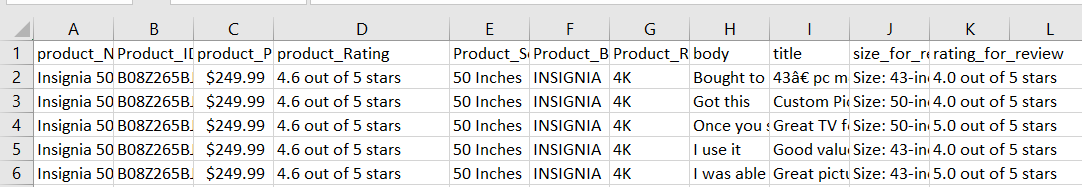
s

**DATA CLEANING**

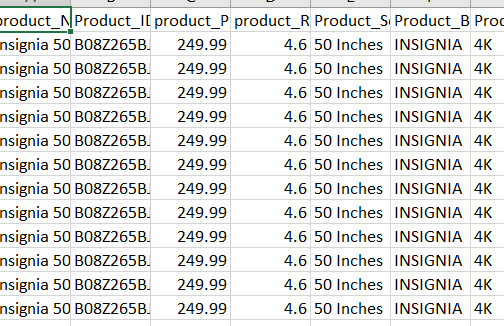
Cleaning data is one of the important parts of any data science project. After data has been pulled from the amazon website various methods are used to clean the data.

* We have removed the extra text after the rating like “out of 5 star” from both Product\_Rating and rating\_for\_review

Before cleaning



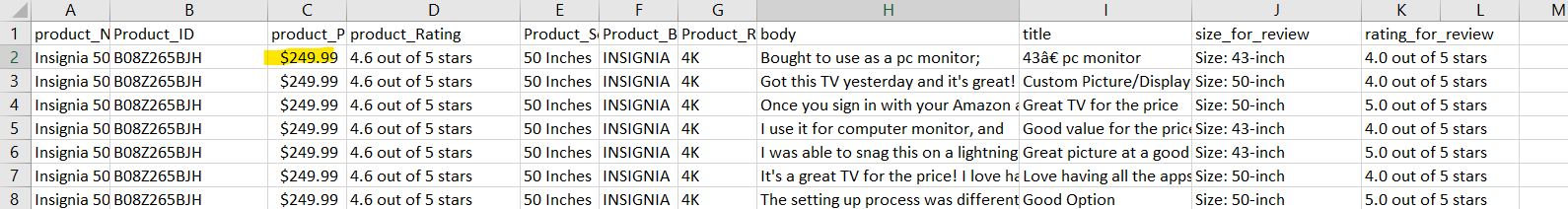
After cleaning



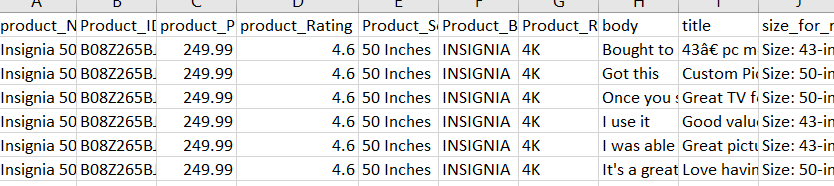
* Removing the $ symbol from price.

When we extracted the data, we see that $ symbol is appended at the start of the product price. We cleaned it for as a part of data cleaning.

Before cleaning



After cleaning:



* Removing products that are not Smart Tv’s

We found that 3 products are not actual Smart Tv’s but are Smart Tv antenna and cables. So, we have decided to remove those product details and their reviews from our final data.

* Removing Inches text

In the Product\_Screensize after the screen size number Inches text is added at the end of it, we have cleaned the data and change the variable datatype from string to Float

* Removing stop words
* Removing punctuation marks
* Removing non-english words and rows from the body column
* Removing numeric values wherever required
* Removing white space characters such as \t, \n

Before cleaning



After cleaning



**DATA TRANSFORMATION**

Making the rating column a numeric. We have converted product\_rating and review\_rating variable from string datatype to float datatype. We are getting extract text after the rating number ex: (4.0 out of 5) so we have deicide to remove the extra text after the float value. This helps us in better analysing and visualizing the data. We found that Amazon fire Tv products don’t have the brand names, we have extracted the brand name from the product name.

**DATA REDUCTION**

We did not create any new attributes or remove any attributes or records other than the missing values. We have just removed the reviews related to product that are not Smart Tv’s.

**DATA CONSOLIDATION**

We have extracted two files one for product details and other for product review details. We have merged both the files using product\_id as the primary key. So, we were able to merge both the files successfully into a single file. We used python padas merge function based on product\_id to merge our data into single file.

**DATA DICTIONARY**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ATTRIBUTE NAME** | **DESCRIPTION** | **DATA TYPE** | **SOURCE** | **Data** |
| Product \_name | Name of the product | String | www.amazon.com | [TV\_Reviews.csv](data/TV\_Reviews.csv) |
| Product\_Id | ASIN number which is unique for any item in amazon website | String | www.amazon.com | [TV\_Reviews.csv](data/TV\_Reviews.csv) |
| Product\_price | Contains the price of the product | float | www.amazon.com | [TV\_Reviews.csv](data/TV\_Reviews.csv) |
| Product\_rating | User rating of the product on amazon website | float | www.amazon.com | [TV\_Reviews.csv](data/TV\_Reviews.csv) |
| Product\_screensize | Describes diagonal Screensize of a product | String | www.amazon.com | [TV\_Reviews.csv](data/TV\_Reviews.csv) |
| Product\_brand | Describes the brand of a selected product. | String | www.amazon.com | [TV\_Reviews.csv](data/TV\_Reviews.csv) |
| Product\_resolution | Resolution defines the amount of horizontal and vertical pixels of tv | String | www.amazon.com | [TV\_Reviews.csv](data/TV\_Reviews.csv) |
| body | Product review or feedback provided by the customers. | String | www.amazon.com | [TV\_Reviews.csv](data/TV\_Reviews.csv) |
| title | Title of User review on products. | String | www.amazon.com | [TV\_Reviews.csv](data/TV\_Reviews.csv) |
| Size\_for\_review | Reviews of customers on products who bought different sizes of same tv. | String | www.amazon.com | [TV\_Reviews.csv](data/TV\_Reviews.csv) |
| Rating\_for\_review | Individual ratings of customers on products. | String | www.amazon.com | [TV\_Reviews.csv](data/TV\_Reviews.csv) |

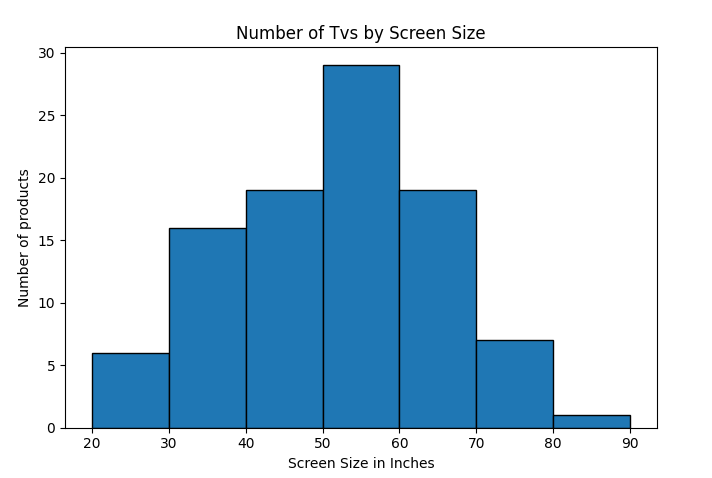
The above table is the data dictionary for the dataset created for this research project.

**TEXT MINING AND SENTIMENT ANALYSIS**

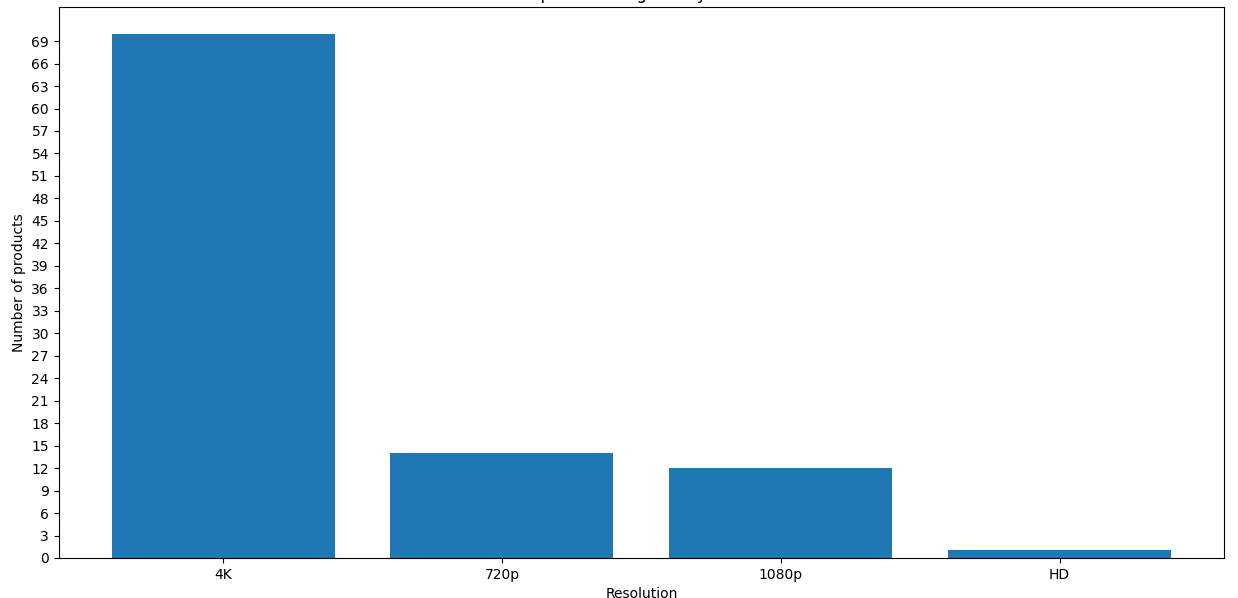
**TOPIC ANALYSIS**

We wanted to group review text together based on the content of their body using Python. These topics can be used to aggregate the content and analyse large volumes of text data by clustering the documents into groups. We have decided to categorize them in to 3 topics.

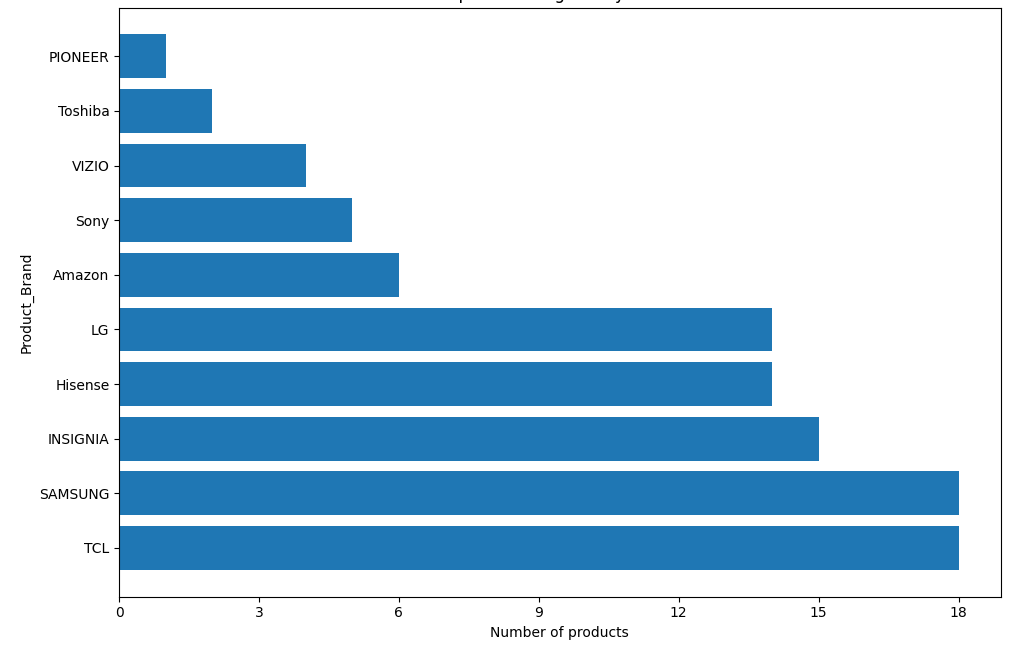
**Visualizations on the extracted data:**



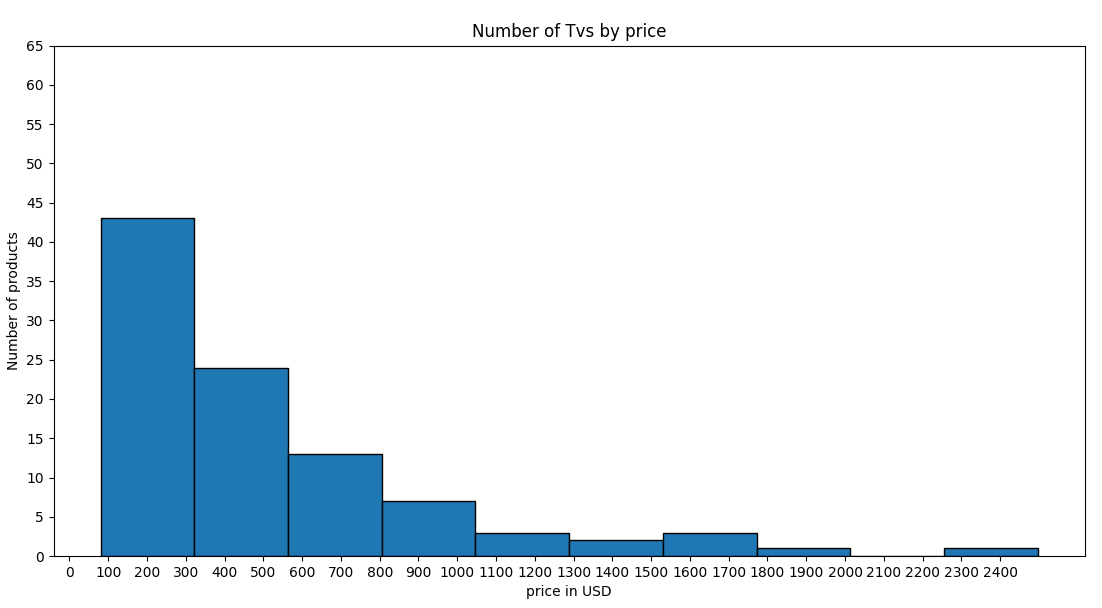
From the above chart we can see that most number of TV’s had a screen size between 50-60 inches.



From the above chart we can find that in the top selling amazon Tv’s majority of the TV’s had a resolution of 4k followed by 720p and 1080p.

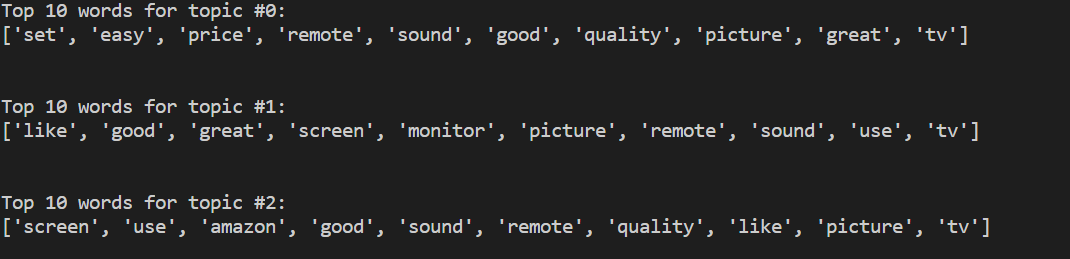


TCL and SAMSUNG had more TV products both are having 18 products each in the top 100 best selling TV’s followed by INSIGNIA and Hisense.



More than 40% of the TV’s had a price below $300 and the number of TV’s in the 100 decreases significantly as the price goes up.

**LDA**



Topic 1 (#0) : Products are at good price for the great quality tv with an easy to operate remote.



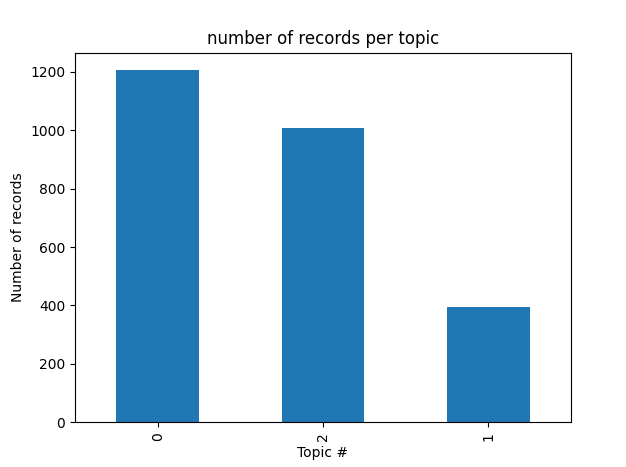
Topic 2 (#1): Products are having good screen /monitor with great sound.



Topic 3 (#2): Tv are with good picture quality and good sound.



The following figure shows the distribution of text descriptions in topics using LDA:



Emotions and analysis.

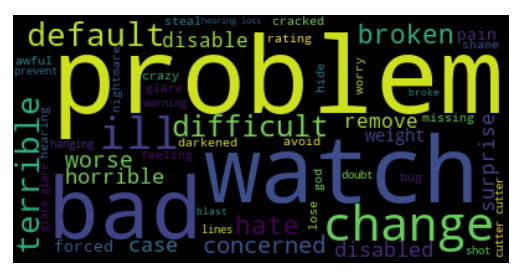
**Trust:**

In trust we see lot of words like happy, excellent, perfect, recommend and good which means customers are trusting the products they bought and would like to recommend it to others.



**Fear:**

In fear we see words like problem, difficult, terrible which makes the customer dissatisfied with the product.

****

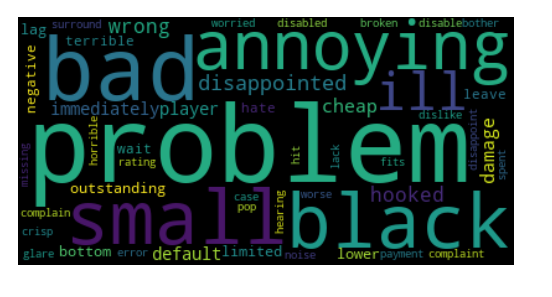
**Positive Cloud:**

In Positive cloud we see word good, love, pretty, perfect which means customers are satisfied with the TV’s they bought and feel good about it.

****

**Negative Cloud:**

Here we see words like problem, bad, annoying which makes the customer unsatisfied with the TV they bought.

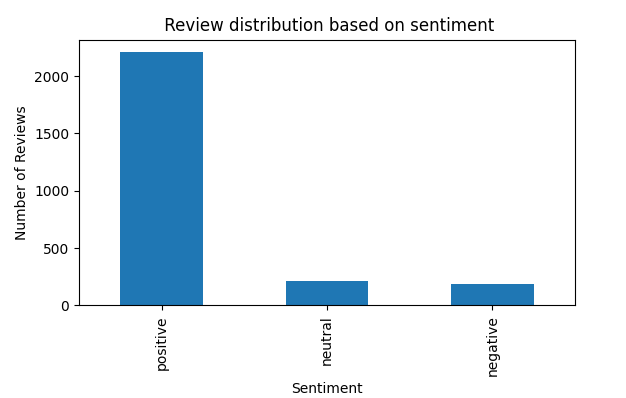
****

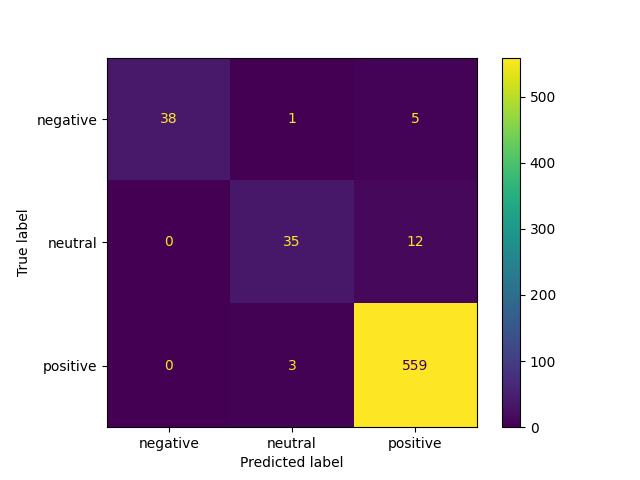
**Classification Model:**

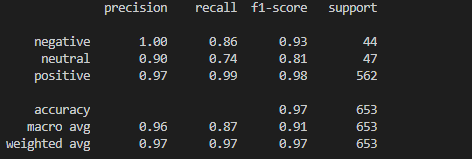
The majority of business challenges never get to the solution stage because of a lack of data labelling. A team of data specialists, using business logic or pre-defined generic packages, and industry experts with domain knowledge often carry out the process of data labelling. They use the categorized data for model training and testing once they have a specific volume of it. The labelling of the data is essential for any classification issue.

We have the same challenge here with the data set. The review column is not a classified data. We used python package “vader” to process the text data and do the sentiment analysis. This package does not consider and review specific terms to do the analysis.

Below is the bar plot that shows the review distribution based on the sentiment.







**Conclusion and Discussion:**

**Summary of data gathering strategies**: While we are scraping the data all the products on amazon website didn’t load completely, we had to send keys to scroll across the webpage to scrape the data and it took lot of time for script to run. We will try to optimize our code so that it takes less time to extract the data.

**Future Implications:** Our research mainly focuses on retail Smart TV consumers; this analysis help the product manufactures to design the TV’s that meet customer expectations. And it further helps to bring new innovations into this fields.

From the analysis we found that most of the reviews are positive that is the reason there are the top 100 best selling Tv’s on Amazon. Most of the customers are buying the Tv’s with the price below three hundred dollars. We see that lot of Tv’s in the top selling had 4k resolution. We also found that a Tv with good sound quality is something that customers are looking for.

**CODE:**

**Data\_cleaning\_2.py**

import pandas as pd

df=pd.read\_table(r"C:\Users\svajjal\OneDrive\project deliverable 1\project-deliverable-1-cowboys\data\Top\_100\_selling\_TV.csv", sep=',')

df.loc[df['product\_Name'].str.contains('Amazon',case=False), 'Product\_Brand'] = 'Amazon'

#print(df)

# updating screen size for amazon tv

size=[]

num\_row=[]

num\_row.extend(df.index[(df['product\_Name'].str.contains('Amazon',case=False) )])

#print(num\_row)

prod\_size=[]

a=df.loc[df['product\_Name'].str.contains('Amazon',case=False)]

#print(a)

size.extend(a['product\_Name'].tolist())

#print(size)

for element in size:

    spl=element.split(' ')

    prod\_size.append(spl[3].rstrip('"'))

#print(prod\_size)

for elem in num\_row:

    i=0

    df.loc[elem, ['Product\_Screensize']] = [prod\_size[i]]

    i=i+1

print(df)

#removing Goove product which is not a tv

df.drop(df[df.Product\_Brand =='Govee'].index, inplace=True)

#Changing rsolution for Amazon tvs

df.loc[df['Product\_Brand']=='Amazon', 'Product\_Resolution'] = '4K'

## Dropping Non TV products

df=df.drop(38)

df=df.drop(60)

print(df)

df.to\_csv(r"C:\Users\svajjal\OneDrive\project deliverable 1\project-deliverable-1-cowboys\data\Top\_100\_selling\_TV.csv",sep=',',index=None)

**textanalysis.py**

import seaborn as sns

import pandas as pd

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.decomposition import LatentDirichletAllocation

from matplotlib import pyplot as plt

import nltk

from nltk.stem import PorterStemmer

nltk.download('punkt')

nltk.download('stopwords')

from textblob import TextBlob

from nltk.corpus import stopwords

import sklearn

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score

import wordcloud as wc

df=pd.read\_table(r"C:\Users\svajjal\Documents\GitHub\project-deliverable-2-cowboys\data\TV\_Reviews.csv", sep=',')

vect = CountVectorizer(max\_df=0.8, min\_df=4, stop\_words='english')

doc\_mat = vect.fit\_transform(df['body'].values.astype('U'))

Latent\_Alloc = LatentDirichletAllocation(n\_components=3, random\_state=15)

Latent\_Alloc.fit(doc\_mat)

for i,topic in enumerate(Latent\_Alloc.components\_):

     print(f'Top 10 words for topic #{i}:')

     print([vect.get\_feature\_names\_out()[i] for i in topic.argsort()[-10:]])

     cloud=[vect.get\_feature\_names\_out()[i] for i in topic.argsort()[-10:]]

     zerocloud=wc.WordCloud(max\_words=50,background\_color='black',contour\_color='black').generate(' '.join(cloud))

     plt.imshow(zerocloud,interpolation='bilinear')

     plt.axis("off")

     plt.show()

     print('\n')

top\_val = Latent\_Alloc.transform(doc\_mat)

top\_val.shape

# Adding topic number  column to df

df['topic#'] = top\_val.argmax(axis=1)

df['topic#'].value\_counts().plot(kind="bar")

plt.title("number of records per topic")

plt.xlabel("Topic #")

plt.ylabel("Number of records")

# plotting the frequency of topic1

plt.show()

reviews=df['body'].values.tolist()

pos\_words=[]

neg\_words=[]

fear\_list=[]

trust\_list=[]

joy\_list=[]

sadness\_list=[]

from nrclex import NRCLex

trust\_fq=[]

fear\_fq=[]

joy\_fq=[]

sadness\_fq=[]

positive\_fq=[]

negative\_fq=[]

for i in range(len(reviews)):

    emotion = NRCLex(reviews[i])

    trust\_list.append(emotion.affect\_frequencies['trust'])

    fear\_list.append(emotion.affect\_frequencies['fear'])

    joy\_list.append(emotion.affect\_frequencies['joy'])

    sadness\_list.append(emotion.affect\_frequencies['sadness'])

    pos\_words.append(emotion.affect\_frequencies['positive'])

    neg\_words.append(emotion.affect\_frequencies['negative'])

for i in range(len(reviews)):

    wordlist=reviews[i].split()

    for word in wordlist:

        word\_emotion=NRCLex(word)

        trust\_value=word\_emotion.affect\_frequencies['trust']

        fear\_value=word\_emotion.affect\_frequencies['fear']

        joy\_value=word\_emotion.affect\_frequencies['joy']

        sadness\_value=word\_emotion.affect\_frequencies['sadness']

        positive\_value=word\_emotion.affect\_frequencies['positive']

        negative\_value=word\_emotion.affect\_frequencies['negative']

        #joy sadness

        if joy\_value > 0:

            joy\_fq.append(word)

        if sadness\_value > 0:

            sadness\_fq.append(word)

        #trust fear

        if trust\_value>0:

            trust\_fq.append(word)

        if fear\_value >0:

            fear\_fq.append(word)

        #pos neg

        if positive\_value > 0:

            positive\_fq.append(word)

        if negative\_value > 0:

            negative\_fq.append(word)

fear\_df=pd.DataFrame()

fear\_df['fear']=fear\_fq

fear\_df['fear'].value\_counts().head(10).plot(kind="bar")

plt.show()

#WordCloud

fearcloud=wc.WordCloud(max\_words=50,background\_color='black',contour\_color='black').generate(' '.join(fear\_fq))

plt.imshow(fearcloud,interpolation='bilinear')

plt.axis("off")

plt.show()

trustcloud=wc.WordCloud(max\_words=50,background\_color='black',contour\_color='black').generate(' '.join(trust\_fq))

plt.imshow(trustcloud,interpolation='bilinear')

plt.axis("off")

plt.show()

positivecloud=wc.WordCloud(max\_words=50,background\_color='black',contour\_color='black',collocations=False).generate(' '.join(positive\_fq))

plt.imshow(positivecloud,interpolation='bilinear')

plt.axis("off")

plt.show()

Negtivecloud=wc.WordCloud(max\_words=50,background\_color='black',contour\_color='black',collocations=False).generate(' '.join(negative\_fq))

plt.imshow(Negtivecloud,interpolation='bilinear')

plt.axis("off")

plt.show()

Review\_details.py

import pandas as pd

import os

import selenium

from selenium import webdriver

from selenium.webdriver.common.by import By

from tkinter.filedialog import SaveAs

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.chrome.service import Service

from time import sleep

ser= Service(r"C:\chrome driver\chromedriver.exe")

sleep(2)

prod\_id\_df = pd.read\_table(r"C:\Users\sriha\OneDrive\Documents\GitHub\project-deliverable-1-cowboys\data\Top\_100\_selling\_TV.csv", sep=',')

#print(prod\_id\_df)

prod\_id\_list = []

#get list of product ids from csv

prod\_id\_list.extend(prod\_id\_df['Product\_ID'].tolist())

print(len(prod\_id\_list))

#url to be replaced with product id

review\_url='https://www.amazon.com/product-reviews/chgprodid/ref=cm\_cr\_getr\_d\_paging\_btm\_prev\_1?ie=UTF8&reviewerType=all\_reviews&pageNumber=1'

no\_of\_Products=1

prod\_id\_1=[]

body\_1 = []

title\_1=[]

size\_1=[]

rating\_1=[]

#iterate through every product id

for a in prod\_id\_list:

    driver = webdriver.Chrome(service=ser)

    review\_url\_upd = review\_url.replace('chgprodid',str(a))

    print(review\_url\_upd)

    driver.get(review\_url\_upd)

    sleep(4)

    driver.maximize\_window()

    sleep(3)

    i=1

    prdid\_cnt=0

    #iterate for 3 review pages

    while(i<4):

        body=driver.find\_elements(By.XPATH,'//span[@class="a-size-base review-text review-text-content"]/span')

        for element in body:

            #sleep(5)

            #print(element.text)

            prdid\_cnt=prdid\_cnt+1

            body\_1.append(element.text)

        title=driver.find\_elements(By.XPATH,'//a[@class="a-size-base a-link-normal review-title a-color-base review-title-content a-text-bold"]/span')

        for element in title:

            #sleep(5)

            #print(element.text)

            title\_1.append(element.text)

        size=driver.find\_elements(By.CSS\_SELECTOR,'a.a-size-mini.a-link-normal.a-color-secondary')

        for element in size:

            size\_1.append(element.text)

        #rating=driver.find\_elements(By.CSS\_SELECTOR,'i.a-icon.a-icon-star.a-star-4.review-rating>span.a-icon-alt')

        rating=driver.find\_elements(By.CSS\_SELECTOR,'div#cm\_cr-review\_list>div>div>div>div:nth-of-type(2)>a:nth-of-type(1)')

        for element in rating:

            rating\_1.append(element.get\_attribute("title"))

        sleep(3)

        if(i==3):

            break

        try:

            next\_btn=driver.find\_element(By.CSS\_SELECTOR,'ul.a-pagination>li:nth-child(2)>a')

            next\_btn.click()

            sleep(1)

        except:

            break

        i=i+1

    driver.quit()

    tbad=1

    while(tbad<prdid\_cnt+1):

        prod\_id\_1.append(a)

        tbad=tbad+1

    no\_of\_Products=no\_of\_Products+1

    #if(no\_of\_Products==21):

        #break

#create dataframe

df=pd.DataFrame()

df['body']=pd.Series(body\_1)

df['Product\_ID']=pd.Series(prod\_id\_1)

df['title']=pd.Series(title\_1)

df['size\_for\_review']=pd.Series(size\_1)

df['rating\_for\_review']=pd.Series(rating\_1)

print(df)

df.to\_csv(r"C:\Users\sriha\OneDrive\Documents\GitHub\project-deliverable-1-cowboys\data\reviews\_total.csv",sep=',',index=None)

Data\_cleaning.py

#rating column

rating=[]

new\_rating=[]

import pandas as pd

df=pd.read\_table(r"C:\Users\svajjal\OneDrive\project deliverable 1\project-deliverable-1-cowboys\data\Top\_100\_selling\_TV.csv", sep=',')

rating.extend(df['product\_Rating'].tolist())

print(len(rating))

for a in rating:

    a=str(a)

    new\_rating.append(float(a[0:3]))

print(len(new\_rating))

df['product\_Rating']=pd.Series(new\_rating)

print(df)

"""

#rating for review

rev\_rating=[]

new\_rev\_rating=[]

rev\_rating.extend(df['rating\_for\_review'].tolist())

print(len(rev\_rating))

for a in rev\_rating:

    if(a!=None):

        new\_rating.append(str(a[0:3]))

    else:

        new\_rating.append(a)

print(len(new\_rev\_rating))

df['rating\_for\_review']=pd.Series(new\_rev\_rating)

"""

#removing $

price=[]

new\_price=[]

price.extend(df['product\_Price'].tolist())

print(len(price))

for a in price:

    if(len(str(a))>1):

        rem=str(a).replace('$','')

        if(rem.find(',')):

            rem=rem.replace(',','')

        print(rem)

        new\_price.append(float(rem))

    else:

        new\_price.append(a)

print(len(new\_price))

df['product\_Price']=pd.Series(new\_price)

#removing inches

inches=[]

new\_inches=[]

inches.extend(df['Product\_Screensize'].tolist())

for a in inches:

    if(len(str(a))>1):

        new\_inches.append(str(a).rstrip(' Inches'))

    else:

        new\_inches.append(a)

df['Product\_Screensize']=pd.Series(new\_inches)

df.to\_csv(r"C:\Users\svajjal\OneDrive\project deliverable 1\project-deliverable-1-cowboys\data\Top\_100\_selling\_TV.csv",sep=',',index=None)

scraper\_product\_details.py

from time import time

from time import sleep

from selenium import webdriver

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.support.ui import WebDriverWait

import os

import pandas as pd

from selenium.webdriver.chrome.service import Service

service\_gecko = Service(executable\_path=r'C:\chrome driver\chromedriver.exe')

driver = webdriver.Chrome(service=service\_gecko)

Amazon\_url = 'https://www.amazon.com/Best-Sellers-Electronics-Televisions/zgbs/electronics/172659'

driver.get(Amazon\_url)

driver.maximize\_window()

from selenium.webdriver.common.by import By

WebDriverWait(driver, timeout=45).until(lambda d: d.find\_element(By.CSS\_SELECTOR,'div#gridItemRoot:nth-of-type(1)>div>div:nth-of-type(2)>div>a>div>img'))

image\_sel='div#gridItemRoot>div#p13n-asin-index-rep>div:nth-of-type(2)>div>a>div>img'

#image\_sel='div#gridItemRoot:nth-of-type(1)>div>div:nth-of-type(2)>div>a>div>img'

price\_sel='div#gridItemRoot>div#p13n-asin-index-rep>div.zg-grid-general-faceout>div>div:nth-of-type(2)>div>div>a>div>span>span'

id\_sel='div#gridItemRoot>div#p13n-asin-index-rep>div.zg-grid-general-faceout>div'

sleep(2)

#from selenium.webdriver.common.action\_chains import ActionChains

product\_Name=[]

product\_ID=[]

product\_Price=[]

product\_Rating=[]

Product\_Screensize=[]

Product\_Brand=[]

Product\_Resolution=[]

height=0

i=0

page\_num=1

while i<51:

    #break for page 2

    if(page\_num==2 and i==50):

        break

    for iter in range(20):

        driver.find\_element(By.TAG\_NAME,'body').send\_keys(Keys.PAGE\_UP)

    #if(i==7):

        #i=i+1

        #continue

    #modifying css selctor based on i value

    image\_sel\_upd=image\_sel.replace('rep',str(i))

    price\_sel\_upd=price\_sel.replace('rep',str(i))

    id\_sel\_upd=id\_sel.replace('rep',str(i))

    for counter in range(35):

        try:

            sleep(2)

            driver.find\_element(By.TAG\_NAME,'body').send\_keys(Keys.PAGE\_DOWN)

            sleep(2)

            if(i==50):

                next\_btn=driver.find\_element(By.CSS\_SELECTOR,'ul.a-pagination>li.a-normal')

                if(next\_btn.size!=0):

                    i=0

                    image\_sel\_upd=image\_sel.replace('rep',str(i))

                    price\_sel\_upd=price\_sel.replace('rep',str(i))

                    page\_num=2

                    next\_btn.click()

                    break

            else:

                find\_img=driver.find\_element(By.CSS\_SELECTOR,image\_sel\_upd)

                if(find\_img.size!=0):

                    break

        except:

            pass

    sleep(5)

    #height=new\_height

    #driver.execute\_script("arguments[0].scrollIntoView();",find\_img)

    prod\_ID=driver.find\_elements(By.CSS\_SELECTOR,id\_sel\_upd)

    if(len(prod\_ID)>0):

        for a in prod\_ID:

            print(a.get\_attribute("ID"))

            product\_ID.append(a.get\_attribute("ID"))

    else:

        product\_ID.append(None)

    prod\_price=driver.find\_elements(By.CSS\_SELECTOR,price\_sel\_upd)

    if(len(prod\_price)>0):

        for a in prod\_price:

            print(a.text)

            product\_Price.append(a.text)

    else:

        product\_Price.append(None)

    prod\_url=driver.find\_element(By.CSS\_SELECTOR,image\_sel\_upd)

    prod\_url.click()

    sleep(4)

    prod\_name= driver.find\_elements(By.CSS\_SELECTOR,'#titleSection>h1>span')

    for a in prod\_name:

        print(a.text)

        product\_Name.append(a.text)

    prod\_rating=driver.find\_elements(By.XPATH,'//div[@id="averageCustomerReviews\_feature\_div"]/div[@id="averageCustomerReviews"]/span/span[@id="acrPopover"][1]')

    if(len(prod\_rating)>0):

        for a in prod\_rating:

            print(a.get\_attribute("title"))

            product\_Rating.append(a.get\_attribute("title"))

    else:

        product\_Rating.append(None)

    prod\_size=driver.find\_elements(By.XPATH,'//table[@class="a-normal a-spacing-micro"]/tbody/tr[@class="a-spacing-small po-display.size"]/td[2]/span')

    if(len(prod\_size)>0):

        for a in prod\_size:

            print(a.text)

            Product\_Screensize.append(a.text)

    else:

        Product\_Screensize.append(None)

    prod\_brand=driver.find\_elements(By.CSS\_SELECTOR,'table.a-normal.a-spacing-micro>tbody>tr.a-spacing-small.po-brand>td:nth-of-type(2)>span')

    if(len(prod\_brand)>0):

        for a in prod\_brand:

            print(a.text)

            Product\_Brand.append(a.text)

    else:

        Product\_Brand.append(None)

    prod\_res=driver.find\_elements(By.CSS\_SELECTOR,'table.a-normal.a-spacing-micro>tbody>tr.a-spacing-small.po-resolution>td:nth-of-type(2)>span')

    if(len(prod\_res)>0):

        for a in prod\_res:

            print(a.text)

            Product\_Resolution.append(a.text)

    else:

        Product\_Resolution.append(None)

    i=i+1

    sleep(1)

    driver.back()

    sleep(1)

print(product\_Name)

print(product\_ID)

print(product\_Price)

print(Product\_Screensize)

print(Product\_Brand)

print(Product\_Resolution)

df=pd.DataFrame()

df['product\_Name']=pd.Series(product\_Name)

df['Product\_ID']=pd.Series(product\_ID)

df['product\_Price']=pd.Series(product\_Price)

df['product\_Rating']=pd.Series(product\_Rating)

df['Product\_Screensize']=pd.Series(Product\_Screensize)

df['Product\_Brand']=pd.Series(Product\_Brand)

df['Product\_Resolution']=pd.Series(Product\_Resolution)

df.to\_csv(r"C:\Users\sriha\OneDrive\Documents\GitHub\project-deliverable-1-cowboys\data\Top\_100\_selling\_TV.csv",sep=',',index=None)

driver.quit()

vaders.py

from nltk.sentiment import SentimentIntensityAnalyzer

from nltk.stem import PorterStemmer

import pandas as pd

from matplotlib import pyplot as plt

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score

import nltk

nltk.download('vader\_lexicon')

df=pd.read\_table(r"C:\Users\svajjal\Documents\GitHub\project-deliverable-2-cowboys\data\TV\_Reviews.csv", sep=',')

portstem = PorterStemmer()

df['stem\_review'] = df['body'].apply(lambda a: " ".join([portstem.stem(text) for text in a.split()]))

print(df)

review\_stemmed=df['stem\_review'].values.tolist()

neg=[]

pos=[]

neu=[]

cmpd=[]

polarity=[]

analyzer = SentimentIntensityAnalyzer()

for rev in review\_stemmed:

    vs = analyzer.polarity\_scores(rev)

    neg.append(vs['neg'])

    pos.append(vs['pos'])

    neu.append(vs['neu'])

    cmpd.append(vs['compound'])

    if(vs['compound']>=0.05):

        polarity.append('positive')

    elif(vs['compound']<=-0.05):

        polarity.append('negative')

    else:

        polarity.append('neutral')

sent\_df=pd.DataFrame()

sent\_df['Review\_stemmed']=review\_stemmed

sent\_df['neg']=neg

sent\_df['pos']=pos

sent\_df['neu']=neu

sent\_df['compound']=cmpd

sent\_df['polarity']=polarity

print(sent\_df)

sent\_df['polarity'].value\_counts().plot(kind='bar')

plt.xlabel("Sentiment")

plt.ylabel("Number of Reviews")

plt.title(" Review distribution based on sentiment")

plt.show()

from sklearn.metrics import ConfusionMatrixDisplay

from nltk.corpus import stopwords

import nltk

nltk.download('stopwords')

reviews\_fts = sent\_df['Review\_stemmed']

stop\_words\_eng=stopwords.words('english')

Tfidf\_vect = TfidfVectorizer(max\_features=2500, min\_df=7, max\_df=0.8,

stop\_words=stop\_words\_eng)

Reviews\_transform\_fts = Tfidf\_vect.fit\_transform(reviews\_fts).toarray()

labels = sent\_df['polarity']

train\_x,test\_x, train\_y, test\_y = train\_test\_split(Reviews\_transform\_fts,

labels, test\_size=0.25, random\_state=0)

rand\_classifier = RandomForestClassifier(n\_estimators=200, random\_state=0)

rand\_classifier.fit(train\_x, train\_y)

predictions = rand\_classifier.predict(test\_x)

con\_matrix = confusion\_matrix(test\_y,predictions)

print(con\_matrix)

ConfusionMatrixDisplay.from\_estimator(rand\_classifier, test\_x, test\_y)

plt.show()

print(classification\_report(test\_y,predictions))

Data\_cleaning\_on\_review.py

import pandas as pd

import nltk

nltk.download('stopwords')

from nltk.corpus import stopwords

reviews\_df=pd.read\_table(r"C:\Users\svajjal\Documents\GitHub\project-deliverable-2-cowboys\data\TV\_Reviews.csv", sep=',')

# Removing rows which does not have reviews

reviews\_df=reviews\_df[reviews\_df['body'].notna()]

#reviews\_df=reviews\_df.reset\_index(drop=True)

print(reviews\_df)

#converting to lower case

reviews\_df['body'] = reviews\_df['body'].apply(lambda x: " ".join(x.lower() for x in x.split()))

#deleting stop words in body column

stop\_words\_eng=stopwords.words('english') +['im']+['fire']

reviews\_df['body'] = reviews\_df['body'].apply(lambda x: " ".join(x for x in x.split() if x not in stop\_words\_eng))

#remove punctuations

rem\_pun = '[^\w\s]'

reviews\_df['body'] = reviews\_df['body'].str.replace(rem\_pun,'')

#removing numbers from body column

num\_pat = '\\b[0-9]+\\b'

reviews\_df['body'] = reviews\_df['body'].str.replace(num\_pat,'')

#rmoving rows with Non english words in body column

reviews\_df=reviews\_df[reviews\_df.body.map(lambda a: a.isascii())]

reviews\_df.to\_csv(r"C:\Users\svajjal\Documents\GitHub\project-deliverable-2-cowboys\data\TV\_Reviews.csv",sep=',',index=None)

datavisualization.py

import pandas as pd

import numpy as np

df=pd.read\_table(r"C:\Users\svajjal\Documents\GitHub\project-deliverable-2-cowboys\data\Top\_100\_selling\_TV.csv", sep=',')

from matplotlib import pyplot as plt

#Top 100 selling Tvs by brand

df\_grpby\_brand=df['Product\_Brand'].value\_counts()

df\_grp\_brand = pd.DataFrame(df\_grpby\_brand).reset\_index()

df\_grp\_brand.columns.values[0] = 'Brand'

df\_grp\_brand.columns.values[1] = 'num\_of\_Tv'

brand=[]

no\_of\_prod=[]

brand.extend(df\_grp\_brand['Brand'].tolist())

no\_of\_prod.extend(df\_grp\_brand['num\_of\_Tv'].tolist())

plt.barh(brand,no\_of\_prod)

plt.tight\_layout()

plt.title("Top 100 selling Tvs by brand")

plt.ylabel('Product\_Brand')

plt.xlabel('Number of products')

plt.xticks(np.arange(0, max(no\_of\_prod) + 2, 3.0))

plt.show()

#pie chart

plt.pie(no\_of\_prod,labels = brand)

plt.show()

#screensize

size\_list=[]

size\_list.extend(df['Product\_Screensize'].tolist())

#print(df['Product\_Screensize'].max)

print(min(size\_list))

bins=[20.0,30.0,40.0,50.0,60.0,70.0,80.0,90.0]

plt.hist(size\_list,bins=bins,edgecolor='black')

plt.tight\_layout()

plt.title("Number of Tvs by Screen Size")

plt.xlabel('Screen Size in Inches')

plt.ylabel('Number of products')

plt.show()

#Price

price\_list=df['product\_Price']

print(min(price\_list))

plt.hist(price\_list,edgecolor='black')

plt.xticks(np.arange(0,max(price\_list),100))

plt.yticks(np.arange(0,70,5))

plt.title("Number of Tvs by price")

plt.xlabel('price in USD')

plt.ylabel('Number of products')

plt.show()

#Resolution

df\_grpby\_resolution=df['Product\_Resolution'].value\_counts()

print(df\_grpby\_resolution)

df\_grp\_res = pd.DataFrame(df\_grpby\_resolution).reset\_index()

df\_grp\_res.columns.values[0] = 'Resolution'

df\_grp\_res.columns.values[1] = 'num\_of\_Tv'

Resolution=[]

no\_of\_prod=[]

Resolution.extend(df\_grp\_res['Resolution'].tolist())

no\_of\_prod.extend(df\_grp\_res['num\_of\_Tv'].tolist())

plt.bar(Resolution,no\_of\_prod)

plt.tight\_layout()

plt.title("Top 100 selling Tvs by brand")

plt.xlabel('Resolution')

plt.ylabel('Number of products')

plt.yticks(np.arange(0, max(no\_of\_prod) + 2, 3.0))

plt.show()