MAJOR RESEARCH PROJECT

ON

Sentiment Analysis of Nike-Dream Further Campaign Using Python

Submitted By:

Ajay Kumar
Enrolment No. - 2K18/MBA/007
MBA (Batch 2018-20)

Under the Guidance of

Asst. Professor Mohit Beniwal

DELHI SCHOOL OF MANAGEMENT

Delhi Technological University Bawana Road Delhi – 110042

CERTIFICATE FROM THE INSTITUTE

This is to certify that the Project Report titled **"Sentiment analysis of Nike-Dream Further Campaign using Python"**, is a bonafide work carried out by **Mr. Ajay Kumar** who is a student of MBA 2018-20 batch at Delhi School of Management, DTU, Delhi. The project is submitted to Delhi School of Management, Delhi Technological University in partial fulfilment of the requirement for the award of the Degree of Masters of Business Administration.

Signature of Guide

Mr. Mohit Beniwal

(Asst. Professor)

Signature of HOD (DSM, DTU)

Dr. Rajan Yadav

(Head of Department)

Seal of Head of Department

(DSM, DTU)

Date:

CANDIDATE'S DECLARATION

I, Ajay Kumar, student of MBA 2018-20 batch of Delhi School of Management, Delhi Technological University, Bawana Road, Delhi – 110042, hereby declare that the major research project report on **"Sentiment analysis of Nike-Dream Further Campaign using Python"** submitted in partial fulfilment of Degree of Masters of Business Administration is the original work conducted by me under the guidance of **Asst. Professor Mohit Beniwal.**

The information and data given in the report is authentic to the best of my knowledge. This report is not being submitted to any other University for award of any degree, diploma and fellowship.

Date:

Ajay Kumar

MBA, 4th Semester, DSM, DTU

Enrolment No. – 2K18/MBA/007

ACKNOWLEDGEMENT

I express my deep gratitude to my Project Guide, Mr. Mohit Beniwal, Delhi School of Management, Delhi Technological University for his enlightening guidance and support throughout by helping me decide the framework for this dissertation and subsequently attending to all my queries. Without his supervision and support this project would not have materialized.

Besides this, the research material published by various government organizations, independent researchers and scholars has helped me gain the basic understanding of concepts involved in the project.

In the end, I would like to extend a word of thanks to the people who guided and kept me motivated throughout this work and helped me whenever asked for.

Ajay Kumar
2K18/MBA/007
MBA (Marketing and IT)

ABSTRACT

Natural Language Processing (NLP) is a hotbed of research in data science these days and one of the most common applications of NLP is sentiment analysis. From opinion polls to creating entire marketing strategies, this domain has completely reshaped the way businesses work, which is why this is an area every data scientist must be familiar with.

Sentiment analysis is the interpretation and classification of emotions (positive, negative and neutral) within text data using text analysis techniques. Sentiment analysis allows businesses to identify customer sentiment toward products, brands or services in online conversations and feedback.

Thousands of text documents can be processed for sentiment (and other features including named entities, topics, themes, etc.) in seconds, compared to the hours it would take a team of people to manually complete the same task.

The key objective of this study is to examine the sentiment of the Nike campaign Dream further which is based on women empowerment. In this project we will use the YouTube as the database. The Nike has posted a video on this campaign and we used the comments of that video as our data.

Comments are collected using YouTube API library followed by the selection of useful features to my task and conversion to CSV. Then, data cleaning is performed like removing URLs, Duplicate values. SentiWordNet lexicon is used to label the sentiment of the C0mments. Steps like Stop words removal, Lemmatizing, Stemming are performed on the text data. Later, WordCloud is used for Data Visualization. After splitting the data, Count Vectorizer and Tfidf Vectorizer are used for mathematical representation of the text. Then, classification algorithms are implemented on the vectors obtained previously.

Contents

Chapter 1: Introduction

Each and every information is very important in the decision making process. After reaching the internet world the user does not get upset about other opinions from newspapers, surveys, opinion pools, consultants as web analytics has started a new system called Opinion Mining using digital social media to create other Finds people's opinions and experiences on networks such as Facebook, reviews, forums, blogs, Twitter, micro-blogs, etc. In fact, according to surveys from 6 to 10 (60%), online shoppers say that by the user Produced customer product reviews have a significant or good impact on their purchasing behavior. . Also data from the 2011 Social Shopping Study indicated that 50% of consumers spend 75% or more of their total shopping time conducting online product research, thus 15% spending 90% or more of their shopping time We do. Another survey by Deloitte Consumer Products Group found that nearly two-thirds (62%) of consumers read consumer-written product reviews online. In fact, a recent study by Deloitte found that "82% of purchase decisions have been directly influenced by reviews". The purpose of this project is to throw off the lime light to determine the spirit of the text, whether positive or negative, which is enhanced by the strength of the polarity. With the explosion of Web 2.0 platforms such as blogs, discussion forums, peer-to-peer networks, and various other types of social media. Consumers have at their disposal a soapbox of unprecedented reach and power by which to share their brand experiences and opinions, positive or negative, about any product or service. As major companies are increasingly feeling these consumer voices can have a huge impact in shaping other consumer opinions and, ultimately, their brand loyalty, their purchasing decisions, and their own brand advocacy. Companies can respond to consumer insights generated through social media monitoring and analytics by modifying their marketing messages, brand positioning, product development, and other activities.

Microblogging websites have evolved to be sources of various types of information. This is due to the nature of the microblog on which people post real-time messages about their opinions on various topics, discuss current issues, complain, and express a positive feeling for products used in daily life We do. In fact, companies that manufacture such products have started stopping these microblogs to get a sense of common sense for their product. Many times, these companies study user reactions and respond to users on the microblog. One challenge is to build technology to detect and summarize a holistic feeling. In this project, we look at one such popular social handle YouTube and receive comments from a special marketing campaign "Dream Forward for Nike's 2019 World Cup" campaign. In this campaign Naik showed women empowerment and powerful spots were used on the line: “Do not change your dream. change the world."

Sentiment analysis is extracting people's perception of a particular issue, brand, scheme, etc., from (text) data (sentiment). It has a wide range of applications, from brand-monitoring, product-review analysis to policymaking. Steps like stop word removal, lemmatizing, steaming are done on text data. Later, WordCloud is used for data visualization. After splitting the data, the count

vectorizer and TFDF vectorizer are used for mathematical representation of the text. Then, nine classification algorithms are applied to the previously obtained vectors.

Literature Review

The article on data cleaning by Omar Elgabri explains the concepts of data because process data cleaning is ultimately very important because the quality of your data improves and in doing so, increases overall productivity. When you clean your data, all old or incorrect information is discarded - leaving you with the highest quality information. Improving data quality is related to data cleaning, detection and removal of data errors and anomalies. Data cleaning plays a major role during the decision making process or data analysis.

• Although data preprocessing is useful, and in many applications, it is necessary to conduct a meaningful data analysis that, if proper techniques are not selected, can result in loss or change of useful information to be discovered during analysis It is possible.

• To perform a meaningful data preprocessing, either the domain expert must be a member of the data analysis team or the domain must be extensively studied before the data is preprocessed. The involvement of a domain expert will provide some useful feedback to validate and validate the use of specialized data preprocessing techniques.

• In most applications, data preprocessing can be iterative. This means that some preprocess techniques, such as data elimination or data selection, can be used in multiple iterations until the best data analysis results are achieved. One of the most important problems in data preprocessing is how we know how much valuable information exists in raw data so that we can ensure that it is protected. This may depend on our definition of data preprocessing. Some may argue that data preprocessing is not a completely "pre" process of data analysis. It requires feedback from the main data analysis process. Ultimately, the ultimate decision of whether someone has done a good job of data preprocessing is to see if "valuable information" is found in the subsequent data analysis process.

The article on "Python Data Science on Starting Tutorials: NLTK", by Swayam Mittal, explains the NLTK library for Python used in this research project. The NLTK module is a huge toolkit designed to help you with a complete natural language processing (NLP) approach. NLTK will provide you with everything from paragraph splitting to sentences, word splitting, identifying part of speech, highlighting topics, and even understanding your machine about the text. In this series, we will address the areas of opinion mining or sentiment analysis.

The main idea is that computers do not understand words directly. It is shocking that humans will not. In humans, the brain breaks down into electrical signals in the brain as a neural group that emits patterns. There are still many unknown things about the brain, but the more we break the human brain into the basic elements, the more basic elements we will find. Well, it turns out that computers store information in a very similar way! If we want to understand the text and understand how humans can understand the text, then we need a way that is as close as possible. In general, computers use numbers to represent everything, but we often use binary signals

directly in programming (true or false, which can be directly converted to 1 or 0 by the presence of an electrical signal. Could (True, 1) or not exist (false, 0)). To do this, we need a way to convert words into numeric or signal patterns. Converting data into something that a computer can understand is called "preprocessing". One of the main forms of preprocessing is to filter out useless data. In natural language processing, useless words (data) are called stop words.

• **Tokenation** - splitting large parts into smaller parts. We can tokenize paragraphs to words and sentences. The process of converting a normal text string into a list of tokens (words we really want).

• **stemming** - removing affix from words and returning the root word. (The stem of the word 'Kama' will be 'Kama'.)

• **Lemmatization** - Word lemmetting is similar to stemming, but the difference lies in the output. Lematized output is a real word, not just a truncated word. For this piece of code to work, you'll need to download the WordNet package for nltk.

The article on "Sentiment Analysis" by Xiao-Li Green talks about pre-processing when the text is muddled, people attempting to express themselves more clearly by adding extravagant punctuation and incorrectly spelling words. Likes to throw in. However, machine learning models cannot cope with text as input, so we have to map characters and words for numerical representations.

Basic pre-processing for the text involves removing non-descriptive characters, intercepting words (a set of very common words such as, a, and, etc.) and converting all words to lowercase. In this example, we also need to remove HTML tags from movie reviews. These steps can be packaged into the following function. A research paper on web sentiment analysis "Theoretical Basic Introduction to Rai Mining" by Pappu Rajan to score positive or negative words using Twitter data. The proposed approach determines the emotion of the text, whether positive or negative, which is enhanced by the strength of the polarity and also acquires important features and the overall sense for each object by calculating a weighted average for all emotions. To analyze. In the text data. For further research, the stock prices of the above companies are collected from the official website of the National Stock Exchange (NSE) for the same period, so compare the overall sentiment of each commodity with its stock prices and prices. Compare predicted results of closure ALM with predicted values using artificial neural networks. There are still many areas for further research, such as the design of efficient algorithms to open mining from positive and negative emotion results.

Research paper on "Sentiment Analysis of Twitter Data" by BoyiXie talks about Sentiment analysis has been handled as a Natural Language Processing task at many levels of granularity. Starting from being a document level classification task (Turney, 2002; Pang and Lee, 2004), it has been handled at the sentence level (Hu and Liu, 2004; Kim and Hovy, 2004) and more recently at the phrase level (Wilson et al., 2005; Agarwal et al., 2009). There are many libraries for text processing and classification-NLTK(comprehensive and a hard learning curve), TextBlob(Easy to

use , suitable for beginners), Spacy(ideal for industry-work), and many more. I choose to go for NLTK because it gives many support libraries to implement many text-processing techniques

Research Methodology

1. **Data Collection**

   Data collection is the process of gathering and measuring information on variables of interest, in an established systematic fashion that enables one to answer stated research questions, test hypotheses, and evaluate outcomes. The data collection component of research is common to all fields of study including physical and social sciences, humanities, business, etc. While methods vary by discipline, the emphasis on ensuring accurate and honest collection remains the same. In this Project the data collection is done using automated techniques of python web scraping from various Websites such as YouTube, LinkedIn, MCI, Facebook, Alexa etc.

2. **Data Cleaning**

   Data cleaning is the process where the data gets cleaned. Data in the real world is normally incomplete, noisy and inconsistent. The data available in data sources might be lacking attribute values, data of interest etc. For example, you want the demographic data of customers and what if the available data does not include attributes for the gender or age of the customers? Then the data is of course incomplete. Sometimes the data might contain errors or outliers. An example is an age attribute with value 200. It is obvious that the age value is wrong in this case. The data could also be inconsistent. For example, the name of an employee might be stored differently in different data tables or documents. Here, the data is inconsistent. If the data is not clean, the data mining results would be neither reliable nor accurate. Data cleaning involves a number of techniques including filling in the missing values manually, combined computer and human inspection, etc. The output of data cleaning process is adequately cleaned data.

3. **Data Integration**

   Data integration is the process where data from different data sources are integrated into one. Data lies in different formats in different locations. Data could be stored in databases, text files, spreadsheets, documents, data cubes, Internet and so on. Data integration is a really complex and tricky task because data from different sources does not match normally. Suppose a table A contains an entity named customer_id where as another table B contains an entity named number. It is really difficult to ensure that whether both these entities refer to the same value or not. Metadata can be used effectively to reduce errors in the data integration process. Another issue faced is data redundancy. The same data might be available in different tables in the same database or even in different data sources. Data integration tries to reduce redundancy to the maximum possible level without affecting the reliability of data.

4. **Data Selection**

"Data mining process requires large volumes of historical data for analysis. So, usually the data repository with integrated data contains much more data than actually required. From the available data, data of interest needs to be selected and stored. Data selection is the process where the data relevant to the analysis is retrieved from the database. In this peoject the relevent data is the comments of users on youtube so we select the relevant data only.

5. **Data Transformation**

"Data transformation is the process of transforming and consolidating the data into different forms that are suitable for mining. Data transformation normally involves normalization, aggregation, generalization etc. For example, a data set available as "-5, 37, 100, 89, 78" can be transformed as "-0.05, 0.37, 1.00, 0.89, 0.78". Here data becomes more suitable for data mining. After data integration, the available data is ready for data mining.

6. **Data Mining**

Data mining is the core process where a number of complex and intelligent methods are applied to extract patterns from data. Data mining process includes a number of tasks such as association, classification, prediction, clustering, time series analysis and so on. In this project we used the NLTK library to extract patterns from the data and perform sentiment analysis.

7. **Pattern Evaluation**

The pattern evaluation identifies the truly interesting patterns representing knowledge based on different types of interestingness measures. A pattern is considered to be interesting if it is potentially useful, easily understandable by humans, validates some hypothesis that someone wants to confirm or valid on new data with some degree of certainty.

8. **Knowledge Representation**

The information mined from the data needs to be presented to the user in an appealing way. Different knowledge representation and visualization techniques are applied to provide the output of data mining to the users.

Objectives

The objectives of the project were as follows :

1. To learn how to use Python and perform data Mining and pre-processing.
2. To learn and use different libraries, APIs and Jupyter notebook.
3. To learn about sentiment analysis and various machine learning algorithms.

Organization of Project Report

This Major Research report gives the documentation of project completed during the 4th semester. There are 4 chapters in this report which are summarized as below -

- Chapter 1: This chapter gives the Introduction of the project and the tools we are going to use for sentiment analysis and the literature review.
- Chapter 2: This chapter talks about the Data collection Process used in the Project.
- Chapter 3: This chapter talks about the Pre-processing of the Data and cleaning of the data using Regex.
- Chapter 4: In this chapter we discuss about the word cloud and different visualization techniques used in sentiment analysis.
- Chapter 5: In this chapter we Implement the machine learning algorithms and fit the model
- Chapter 6: Results
- Chapter 7: Conclusion

Chapter 2: Data Collection

Introduction

Sentiment analysis now requires text data that expresses people's sentiment about the Nike campaign. Some of the available options are using the YouTube API, Facebook API, Twitter API, and scraping data from web pages. I consider YouTube to be a large psychological database with the YouTube API "Dream further | Nike".

Comments for projects are collected from YouTube using the YouTube API. The API stands for Application Programming Interface.

The API can generate both internal and external types of large-scale values. Managing and processing data is one of the important factors in business management, and every company has built IT systems. As the size of data is increasing rapidly, however, there is a limit to all data being encountered through traditional IT systems. In this sense, implementing APIs can be a solution with better efficiency and security. It can break down barriers between systems, which simplifies work processes, enables interoperability between organizations and higher preservation of data.

The external properties of the API are also hardcore. If a company opens its API publicly or with additional fees, it can provide new services and gain potential customers in their favor. Customers can experience the high level of services already available. By offering API services, third-party developers can create whole new types of products that companies had never thought of. For example, the most popular API among developers, Google Maps, did not previously expect to attract that effect. By applying those figures to real estate and various other fields, developers brought a lot of value and property back to Google.

Nowadays the number of APIs is continuously increasing, and this trend will continue or even more. Now we can say that the implementation and management of a company's API is one of the important factors for its competitive and strategic values. The YouTube Application Programming Interface (YouTube API) allows developers to access video statistics and YouTube channel data via two types of calls, REST and XML-RPC. Google describes YouTube API resources as "APIs and tools that allow you to bring the YouTube experience to your webpage, application, or device. The Player and Player API section identities are ways you can provide your users with your application You can watch YouTube videos and control the playback experience. With an embedded YouTube player, you can integrate the YouTube video playback experience directly into your web page or application. You can use the player to customize the format of the player. Can use parameters, and you can also use the Player API to control the player directly from your web page or app.

Google provides a large set of API’s for the developer to choose from. Each and every service provided by Google has an associated API. Being one of them, Youtube Data API is very simple to use provides features like –

- Search for videos

- Handle videos like retrieve information about a video, insert a video, delete a video etc.
- Handle Subscriptions like lists all the subscriptions, insert or delete a subscription.
- Retrieve information about comments like replies to a specific comment identified by a parentId etc.

Jupyter Notebook

Jupyter Notebook Interface

1. Running Jupyter - On Windows, you can run Jupiter via the shortcut Anaconda Add to your Start Menu, which will open a new tab in your default web browser that should show something. This is a notebook dashboard, specifically designed for managing your Jupiter notebooks. Think of it as a launchpad for discovering, editing, and creating your notebook. Be aware that the dashboard will only give you access to files and subfolders within Jupyter's start-up directory; However, the start-up directory can be changed. It is possible to start a dashboard on any system via a command prompt (or terminal on a Unix system) by entering the command jumper notebook; In this case, the current working directory will be the start-up directory.

2. ipynb File – It will be useful to understand what this file really is. Each. ipynb file is a text file that describes the contents of your notebook in a format called JSON. Each cell and its contents, including image attachments that have been converted into strings of text, is listed therein along with some metadata. You can edit this yourself — if you know what you are doing! — by selecting “Edit , Edit Notebook Metadata” from the menu bar in the notebook.You can also view the contents of your notebook files by selecting “Edit” from the controls on the dashboard, but the keyword here is “can“; there’s no reason other than curiosity to do so unless you really know what you are doing.

3. Keyboard Shortcuts –

   – Toggle between edit and command mode with Esc and Enter, respectively.

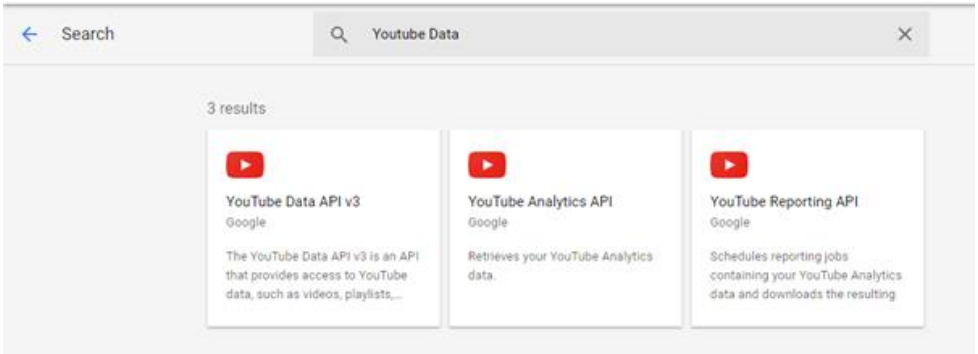
   – Once in command mode:

- Scroll up and down your cells with your Up and Down keys.
- Press A or B to insert a new cell above or below the active cell.
- M will transform the active cell to a Markdown cell.
- Y will set the active cell to a code cell.
- D + D (D twice) will delete the active cell.
- Z will undo cell deletion.
- Hold Shift and press Up or Down to select multiple cells at once.
- With multiple cells selected, Shift + M will merge your selection.
- Ctrl + Shift + -, in edit mode, will split the active cell at the cursor.
- You can also click and Shift + Click in the margin to the left of your cells to select them.

4. Kernels- A kernel runs behind each notebook. When you run a code cell, that code is executed within the kernel and any output is displayed back in the cell. The state of the kernel persists over time and between cells - this is related to the document and not as individual cells.

5. Save and Checkpoint - Each time you create a new notebook, a checkpoint file is created as well as your notebook file; It will be located within a hidden subdirectory of your save location called ipynb_checkpoint and is also an ipynb file. By default, Jupiter will automatically save your notebook to this checkpoint file every 120 seconds without making changes to your primary notebook file. When you "save and checkpoint", the notebook and checkpoint files are updated. Therefore, the checkpoint enables you to recover your saved work in the event of an unexpected problem. You can check the file at the checkpoint from the menu, "Return to checkpoint."

6. Steps to enable the API and start using it



YOUTUBE API Website

Create New Project, Enable API and Create Credentials:

In this step we will create a project and will enable the API.

• Go to Google Developers Console and click on Sign in at the top right corner of the page. Sign in using valid Google account credentials. If you do not have a Google account, set up an account first and then use the details to sign in on the Google Developers homepage.

• Now go to the developer dashboard and create a new project.

• Click the Enable API option.

• In the search field, search for the YouTube Data API and select the YouTube Data API option that appears in the drop-down list.

• You will be redirected to a screen that says information about the Youtube Data API with two options: enable and TRY API

• Click the Enabled option to get started with the API.

• In the sidebar under APIs and Services, select Credentials.

• In the Credentials tab, select the Credentials drop-down list, and choose the API key.

There are two types of Credentials: API Key and OAuth. OAuth provides you with Client Id and a Secret Key in the form of a .json file. OAuth is generally used where authorization is required like in the case of retrieving liked videos of a user. So, for the rest cases where authorization is not required like searching for the videos using a keyword or for searching for the related videos etc we will be using API Key.

Installation

Google API client for python can be installed using simple pip command:

- pip install –upgrade google-api-python-client

```
api_key="AIzaSyDpe7Dy62lEsz4SpbB25ZRIckXRFre80Bw"
youtube=build("youtube","v3",developerKey=api_key)
#fetch youtube comments for a particular video
video_id = "hOVkEHADCg4"
nextPageToken=None
result=[]
while(1):
    try:
        request = youtube.commentThreads().list(videoId = video_id,part = "id, snippet"
,maxResults=40,pageToken=nextPageToken)
        response = request.execute()
        nextPageToken=response["nextPageToken"]
        result.append(response.get("items",[]))
    except:
        break
print(result[0][0])
```

```
{'kind': 'youtube#commentThread', 'etag': '"nxOHAKTVB7baOKsQgTtJIyGxcs8/6v
FQRDr--dFtNsHrgO__34mF6cE"', 'id': 'UgxrwVa2t029RrpQtIF4AaABAg', 'snippe
t': {'videoId': 'hOVkEHADCg4', 'topLevelComment': {'kind': 'youtube#commen
t', 'etag': '"nxOHAKTVB7baOKsQgTtJIyGxcs8/5ub5Wb_IM2hjfAqEcxxaoXHDjT8"',
'id': 'UgxrwVa2t029RrpQtIF4AaABAg', 'snippet': {'authorDisplayName': 'Empo
wered Seal', 'authorProfileImageUrl': 'https://yt3.ggpht.com/a/AATXAJzYSg8
ZdksTlfO8GtJjRiubRtCABJs4XuFRUg=s48-c-k-c0xffffffff-no-rj-mo', 'authorChan
nelUrl': 'http://www.youtube.com/channel/UCGr-R2zPrKrfVA7Yu7Tal2g', 'autho
rChannelId': {'value': 'UCGr-R2zPrKrfVA7Yu7Tal2g'}, 'videoId': 'hOVkEHADCg
4', 'textDisplay': 'Top 10 things that will never happen', 'textOriginal':
'Top 10 things that will never happen', 'canRate': True, 'viewerRating':
'none', 'likeCount': 0, 'publishedAt': '2020-04-22T14:52:21.000Z', 'update
dAt': '2020-04-22T14:52:21.000Z'}}, 'canReply': True, 'totalReplyCount':
0, 'isPublic': True}}
```

Import The Youtube API client Using Jupyter Notebook

As you can see from the image that we got the API key and used youtube.commentTreads() function to fetch all the comments present on the video having video id mentioned above. The we used the Print result command to print the result we got from the API and it is in JSON Format as mentioned above.

Chapter 3: Pre-processing of YouTube Data

Introduction

In any machine learning process, data preprocessing is the step in which data is transformed, or encoded, to bring it to a state that now the machine can easily parse it. In other words, the characteristics of the data can now be easily interpreted by the algorithm. A dataset can be viewed as a collection of datasets, often referred to as records, points, vectors, patterns, events, cases, samples, observations, or entities. Data objects are characterized by a number of attributes, which capture the basic characteristics of an object, such as the mass of a physical object or the time at which an event occurs, etc. .. Features often include variables, attributes, fields Called as, qualities, or dimensions. Features can be:

• hierarchical: attributes whose values are derived from a defined value. For example, days in the week: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday is a category because its value is always taken for this set. Another example could be a Boolean set: true, false

• Numeric: attributes whose values are constant or integer-valued. They are represented by numbers and possess most of the properties of numbers. For example, how many steps you take in a day, or the speed at which you are driving your car.

Because data is often taken from multiple sources that are usually not very reliable and that too in various formats, more than half of our time is spent dealing with data quality issues when working on a machine learning problem. It is easy to expect that the data will be correct. Problems can occur due to human error, range of measuring devices, or flaws in the data collection process. Let's learn about some of them and ways to deal with them:

1. Missing values: It is very common to have missing values in your dataset. This may have occurred during data collection, or may have been due to some data validation rule, but missing values should be taken into account regardless.

Eliminate rows with missing data: Simple and sometimes effective strategy. Many items fail if they do not have values. If a feature has mostly missing values, then that feature can also be eliminated. Estimate Missing Values: If only appropriate percentage values are missing, we can also run a simple interpolation method to fill those values. However, the most common way of dealing with missing values is to fill them with the mean, average, or mode value of the corresponding feature.

2. Inconsistent values: We know that data can contain inconsistent values. Most probably we have already faced this issue at some point. For example, the 'Address' field contains 'Phone Number'. This may be due to human error or perhaps the information was incorrect during the handwritten scan. Therefore it is always advisable to do data evaluation such as knowing what the data type should be and whether it is the same for all data objects.

3. Duplicate Values: A dataset may include data objects that are duplicates of each other. This can happen when it is said that the same person submits the form more than once. The term

duplicate is often used to refer to the process of dealing with duplicates. In most cases, duplicates are removed so as not to give advantage or bias to that particular data object while running the machine learning algorithm.

Dimensionality Reduction

Most real-world datasets have a large number of features. For example, consider an image processing problem, we may have to deal with thousands of attributes, also called dimensions. As the name suggests, dimensionality reduction is intended to reduce the number of features - but only by selecting a sample of features from the feature-set, which is something else - feature subset selection or simply feature selection. Conceptually, the dimension refers to the number of geometric planes that are contained in the dataset, which may be so high that it cannot be visualized with pen and paper. The greater the number of such planes, the greater the complexity of the dataset.

Formation of Data Frame using Pandas.

Pandas Data Frames make it easier to manipulate your data by choosing or changing columns and indices to manipulate your data. Pandas is a popular Python package for data science, and with good reason: it's o ers powerful, expressive, and that's external data structures that make data manipulation and analysis easier, among many other things. The DataFrame is one of these structures. We get the JSON file related part for sentimental analysis, only required to do the comment text so that we use for loop and get individual comments and store it in the video variable. Then we converted that video into a variable data frame using a library of pandas. The output of df.head () is shown where the top 5 rows are printed and showing the comments.

```
videos=[]
for res in result:
    for res1 in res:
        videos.append(res1["snippet"]["topLevelComment"]["snippet"]["textDisplay"])
```

```
In [5]:
```

```
df=pd.DataFrame(data=videos,columns=["full_text"])
```

```
In [6]:
```

```
df.to_excel("output_youtube.xlsx")
```

Extract Relevant Data from the Raw imported Data

```
df.head(10)
```

Out[70]:

	full_text
0	Top 10 things that will never happen
1	Who's the girl?
2	nike: change the world by playing a game...and...
3	This has now become my favourite advert ever x...
4	Beautiful
5	I got goosebumps watching it. Too good. . Nik...
6	I love it!!!!
7	Little girl means Nike or God of fortune??
8	so i wanna start football now
9	It is sad that you advertise this , while your...

Output after extraction of relevant data

Data cleaning

In this step we created a function called cleandf() which takes a Data frame as input and clean the files as per the requirements. The steps of cleaning the data frame are as follows.

- Remove Duplicate Rows
- Remove the unnecessary text and symbols using RE function
- We replace all these symbols with a whitespace character.
- The re.sub() function searches for a pattern and replaces with the text we specify.

```
def clean_df(df_copy):

    #DROPS
    #CHOOSE EITHER TO DROP ALL-ROW DUPLICATES OR FULL_TEXT DUPLICATES
    df_copy=df_copy.drop_duplicates(['full_text']) #3377 left after this
    df_copy=df_copy.reset_index(drop=True)
    #df_copy=

    # BASIC CLEANING FUNCTION
    for i in range(len(df_copy)):
        txt = df_copy.loc[i]["full_text"]
        txt=re.sub(r'@[A-Z0-9a-z_:]+','',txt)#username-tags
        txt=re.sub(r'^[RT]+','',txt)#RT-tags
        txt = re.sub('https?://[A-Za-z0-9./]+','',txt)#URLs
        txt=re.sub('br',"",txt)
        txt=re.sub("[^a-zA-Z]", " ",txt)#hashtags
        df_copy.at[i,"full_text"]=txt
```

Cleaning the data

POS-Tagging and Sentiment labelling

We are done with the basic cleaning part of text data. In the ML algorithms that we are going to implement 'full_text' of the comments acts as a predictor variable(other variables that can be used are reply-count, favorite-count if we want to predict the impact of a comment, but that's

not a part of our task currently). As it is evident, we need to create target variables (sentiment scores) for our data. For this purpose, we use SentiWordNet. SentiWordNet is an enhanced lexical resource explicitly devised for supporting sentiment classification and opinion mining applications. It has a large corpus of POS-tagged English words along with their sentiment.

```
    li_swn_pos=[]
    li_swn_neg=[]
    missing_words=[]
    for i in range(len(df_copy.index)):
        text = df_copy.loc[i]['full_text']
        tokens = nltk.word_tokenize(text)
        tagged_sent = pos_tag(tokens)
        store_it = [(word, map_tag('en-ptb', 'universal', tag)) for word, tag in tagged
_sent]
        #print("Tagged Parts of Speech:",store_it)

        pos_total=0
        neg_total=0
        for word,tag in store_it:
            if(tag=='NOUN'):
                tag='n'
            elif(tag=='VERB'):
                tag='v'
            elif(tag=='ADJ'):
                tag='a'
            elif(tag=='ADV'):
                tag = 'r'
            else:
                tag='nothing'

            if(tag!='nothing'):
                concat = word+'.'+tag+'.01'
                try:
                    this_word_pos=swn.senti_synset(concat).pos_score()
                    this_word_neg=swn.senti_synset(concat).neg_score()
                    #print(word,tag,':',this_word_pos,this_word_neg)
                except Exception as e:
                    wor = lem.lemmatize(word)
                    concat = wor+'.'+tag+'.01'
                    # Checking if there's a possiblity of lemmatized word be accepted i
nto SWN corpus
                    try:
                        this_word_pos=swn.senti_synset(concat).pos_score()
                        this_word_neg=swn.senti_synset(concat).neg_score()
                    except Exception as e:
```

POS tagging and sentiment labelling

SWN gives pos_score and neg_score for each word as in the above case. The higher the pos_score, the more positive is the word. We use a simple linear summation of these scores(We add pos_score of all the words in a comment to form a pos_total and in a similar way, we obtain neg_total. Then we add these two to obtain sent_total) and label a sentence as positive(1) if it(sent_total) is greater than 0, negative(-1) if it is less than 0 and neutral(0), otherwise. As you may have figured out, we need to find POS-tags(Parts of Speech-tags) of our comments in order to use SWN. The pos_tag feature of nltk helps us to tag the text, but it is not simplified(Besides noun, pronoun, adjective, verb it also tags text with many more tags like adposition, conjunction, determiner, etc.,). But SWN accepts only simplified tags. So, we try to simplify the tags using map_tag function from nltk library.

Stop Words Removal

Stop words are commonly used words such as', 'the', 'etc.' In our context, they do not add too much weight to emotion. Also, they increase the mobility of data. So we remove the word stop using the nltk library (in the next part). Before feeding this textual data to the ML algorithm, we transform it into vectors (in later parts of the article). And we don't want to have different (vector) values for the same words. We expect the same values for all these terms -, distribution ', distribution', same distribution ', they distribution' because they give essentially the same meaning. This is better achieved (assigning the same values to similar words) if we use the Lemmatizing and Stemming nltk modules.Lemmatizing and Stemming

Stemming reduces the words to their stems. Stemming algorithms are majorly rule-based ones. For instance, they reduce all the above-'distribute' words to 'distribut'. A lemmatizer does the same thing as Stemmer but keeping the linguistics of the word in context. For instance, all the above words will be condensed to 'distribute'."

```
    #3-rd for loop
    #LEMMATIZING,STEMMING,STOP-WORDS
    for i in range(len(df_copy.index)):
        text = df_copy.loc[i]['full_text']
        tokens = nltk.word_tokenize(text)
        tokens = [word for word in tokens if word not in stop_words]

        for j in range(len(tokens)):
            tokens[j] = lem.lemmatize(tokens[j])
            tokens[j] = pstem.stem(tokens[j])

        tokens_sent=' '.join(tokens)
        df_copy.at[i,"full_text"] = tokens_sent

    df_copy.insert(1,"pos_score",li_swn_pos,True)
    df_copy.insert(2,"neg_score",li_swn_neg,True)
    df_copy.insert(3,"sent_score",li_swn,True)



    return df_copy

st=time.time()
df_copy = clean_df(df_copy)
end=time.time()
```

Lemmatizing and Stemming

It is worth noting that the above three cleaning processes(Stop words Removal, Stemming, Lemmatizing) are not performed before(sentiment labeling) because doing so will cause irregularities in sentiment detection. The words like 'distribut', 'cleanin' can't be found in SWN's corpus of English words, thus SWN fails to detect positive and negative score of these words. This makes comments from only a part of the dataset labeled with a sentiment score. We also convert all comments to lower case using nltk library. Then we find the sentiment of the comments by the method, mentioned above in the introduction. After obtaining sentiments from SWN, we add three more columns to our CSV- 'pos_score', 'neg_score', 'sent_score'. The 'pos_score' of a comments is the net positive score of all words in a comments and the similar goes with 'neg_score'.

```
df_copy.head()
```

Out[19]:

	full_text	pos_score	neg_score	sent_score
0	op thing never happen	0.000	0.625	-1
1	who girl	0.000	0.000	0
2	nike chang world play game buy everi last one ...	0.000	0.250	-1
3	becom favourit advert ever xoxoxxoxooxxoxooxooox	0.125	0.000	1
4	beauti	0.750	0.000	1

Output after preprocessing

Chapter 4: Data Visualization

Introduction

Data visualization refers to presenting data in graphical or graphical form, such as pie charts. This allows viewers to identify patterns more quickly. Interactive visualizations allow decision-makers to drill down through layers of detail. This approach changes so that users can review the facts behind the analysis. Here are ways that data visualization affects decision making and change organizations.

- **Improved Insight**

data visualization can provide insights that traditional descriptive statistics cannot. An accurate example of this is Anscombe's quartet, created in 1973 by Francis Anscombe. This illustration includes four different datasets with approximately equal variance, mean, correlation between X and Y coordinates, and linear regression lines. However, the patterns are clearly different when plotted on a graph. Below, you can see that the linear regression model graph will apply to one and three, but the polynomial regression model graph would be ideal for two. This depiction sheds light on why it is important to visualize data and not just rely on descriptive statistics.

- **Fast decision making**

Companies that can gather on their data and act faster will be more competitive in the market because they can make informed decisions sooner than the competition. Speed is the key, and data visualization helps in understanding vast amounts of data by implementing visual representation of data. This visualization layer typically sits on top of a data warehouse or data lake and allows users to search and locate data in a self-serving manner. This not only promotes creativity, but it reduces the need for IT to allocate resources to create new models.

For example, says a marketing analyst, who works on 20 different advertising platforms and internal systems need to quickly understand the effectiveness of marketing campaigns. A manual way to do this would be to go to each system, draw a report, combine the data, and then analyze in Excel. The analyst will then need to look at a swarm of matrices and attributes and have difficulty drawing conclusions. However, modern business intelligence (BI) platforms will automatically connect to data sources and layer on data visualization so that analysts can easily slice and dice the data and quickly come to conclusions about marketing performance.

Tools Used for visualization in this project

Matplotlib

Data visualization is a key skill for aspiring data scientists. Matplotlib makes it easy to create meaningful and insightful plots. Here, you'll learn how to build various types of plots, and customize them to be more visually appealing and interpret-able. Matpoltlib is a very popular Python library for data visualization. Like Pandas, it is not directly related to Machine Learning. It

particularly comes in handy when a programmer wants to visualize the patterns in the data. It is a 2D plotting library used for creating 2D graphs and plots. A module named pyplot makes it easy for programmers for plotting as it provides features to control line styles, font properties, formatting axes, etc. It provides various kinds of graphs and plots for data visualization, viz., histogram, error charts, bar chats, etc.

Textblob

TextBlob is a python library and offers a simple API to access its methods and perform basic NLP tasks. A good thing about TextBlob is that they are just like python strings. So, you can transform and play with it same like we did in python. Below, I have shown you below some basic tasks. Don't worry about the syntax, it is just to give you an intuition about how much-related TextBlob is to Python strings. TextBlob is a Python (2 and 3) library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.

```
from textblob import TextBlob
count_total=0
count_pos=0
count_neg=0
count_neut=0

li_tb = []
for i in range(len(df_copy.index)):
    sent = TextBlob(str(df_copy.iloc[i]['full_text']))
    if(sent.sentiment.polarity>0):
        count_pos=count_pos+1
        count_total=count_total+1
        li_tb.append(1)
    elif(sent.sentiment.polarity<0):
        count_neg=count_neg+1
        count_total=count_total+1
        li_tb.append(-1)
    else:
        li_tb.append(0)
        count_neut+=1

        count_total=count_total+1


#         print(df.loc[i]['full_text'])
#         print(sent.sentiment)
print("Total tweets:",len(df.index))
print("Total tweets with sentiment:",count_total)
print("positive tweets:",count_pos)
print("negative tweets:",count_neg)
print("neutral tweets:",count_neut)
```

```
Total tweets: 1600
Total tweets with sentiment: 1582
positive tweets: 469
negative tweets: 176
neutral tweets: 937
```

Code for TextBlob and Matplotlib

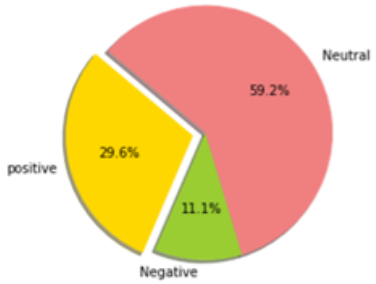
As we can see from the Image where we used text blob to count the number of positive, negative and neutral comments and then we plotted a PIE chart using the matplotlib library

In [65]:

```
# Data to plot
labels = 'positive', 'Negative', 'Neutral'
sizes = [count_pos, count_neg, count_neut]
colors = ['gold', 'yellowgreen', 'lightcoral']
explode = (0.1, 0, 0)  # explode 1st slice

# Plot
plt.pie(sizes, explode=explode, labels=labels, colors=colors,
autopct='%1.1f%%', shadow=True, startangle=140)

plt.axis('equal')
plt.show()
```



Plot of PIe chart using text blob and matplotlib

Wordcloud

Word clouds (also known as wordl, word collage, or tag cloud) are visual representations of words that give more prominence to words that appear more frequently. For Mentmeter Word Clouds, words that are most frequently added by audience members using their smartphones. This type of visualization can help presenters quickly gather data from their audience, uncover the most common answers, and present the data in a way that everyone can understand. We use

WordCloud to represent most words in each sense. To that end, we add the text of all comments that contain a positive label to form a string and represent this string as pos_string.

We follow the same method for negative and neutral text.

```
pos_text=""
neg_text=""
neut_text=""

for i in range(len(df_copy.index)):
    if(df_copy.loc[i]["sent_score"]==1):
        pos_text+=df_copy.loc[i]["full_text"]
    elif(df_copy.loc[i]["sent_score"]==-1):
        neg_text+=df_copy.loc[i]["full_text"]
    else:
        neut_text+=df_copy.loc[i]["full_text"]

list_text = [pos_text,neg_text,neut_text]


for txt in list_text:
    word_cloud = WordCloud(width = 600,height = 600,max_font_size = 200).generate(txt)
    plt.figure(figsize=(12,10))# create a new figure
    plt.imshow(word_cloud,interpolation="bilinear")
    plt.axis("off")
    plt.show()
```

Code for Word cloud

Plot of POS_text word cloud

Plot of NEG_text word cloud

Plot of NEUT_text word cloud

Chapter 5:Training the Models

Splitting the data

We split our data into training, validation and test sets using train_test_split function of model_selection module from nltk library. The division is as follows — 90% of training data, 5% validation data, and 5% test data. More proportion of data is for training, keeping in mind the smaller number of Comments.

```
import sklearn
from sklearn.model_selection import train_test_split
SEED=4
x = df_copy.full_text
y = df_copy.sent_score
x_train,x_val_test,y_train,y_val_test = train_test_split(x,y,test_size=0.1,random_state
=SEED)
x_val,x_test,y_val,y_test = train_test_split(x_val_test,y_val_test,test_size=0.5,random
_state=SEED)
```

Splitting of data using test train split

Vectorizing the text of Comments

Before we apply different ML text classifiers, we need to convert the text data into vectors. This is important because algorithms expect data in some mathematical rather than textual form. We apply two vectorizers from Scalar - Calculate Vector Cityizer and Tuffed Cleaner. Both of these fall under the bag-of-words model. The compute vectorizer counts the number of times a word appears in the document (in each comment) and uses this value as its weight. To visualize, all comments in a vector (after vectoring) have all different words and each line contains comments. The counter vectorizer gives us a sparse matrix (filled with many zeros) as a vector.

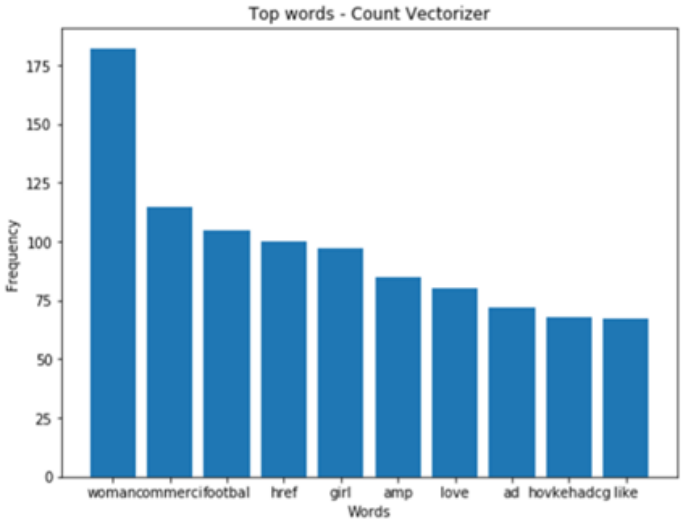
```
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(decode_error='ignore',lowercase=False,max_features=11)
x_traincv=cv.fit_transform(x_train.values.astype('U'))
top_sum=x_traincv.toarray().sum(axis=0)
top_sum_cv=[top_sum]#to let pandas know that these are rows
columns_cv = cv.get_feature_names()
x_traincvdf = pd.DataFrame(top_sum_cv,columns=columns_cv)


import operator
dic = {}
for i in range(len(top_sum_cv[0])):
    dic[columns_cv[i]]=top_sum_cv[0][i]
sorted_dic=sorted(dic.items(),reverse=True,key=operator.itemgetter(1))
print(sorted_dic[1:])
bins = [w for w,v in sorted_dic][1:]#slicing to delete the first swachh bharat
freq = [v for w,v in sorted_dic][1:]
from matplotlib import pyplot as plt

plt.figure(figsize=(8,6))
plt.bar(bins,freq)
plt.xlabel('Words')
plt.xlabel('Words')
plt.ylabel('Frequency')
plt.title('Top words - Count Vectorizer')
plt.show()
```

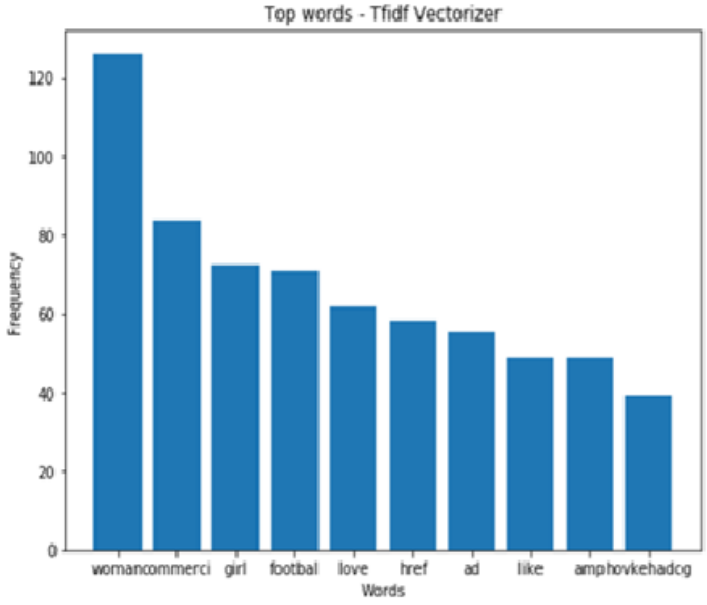
Vectorizing the text of comments

```
[('woman', 182), ('commerci', 115), ('footbal', 105), ('href', 100), ('gir
l', 97), ('amp', 85), ('love', 80), ('ad', 72), ('hovkehadcg', 68), ('lik
e', 67)]
```



output

TF-IDF stands for "term frequency-inverse document frequency". The weight assigned to a token is the product of the term(word) frequency and the inverse of document frequency of the word. Thus, if a word appears more often in the comments, it is assigned less importance(IDF-weight) in this vectorizer.



output term frequency-inverse document frequency

Implementing classifiers from sklearn

After vectorizing our comments, we are all set to implement classification algorithms. The classifiers we are going to implement are-Naive Bayes models(MultinomialNB, BernoulliNB,), Linear models (LogisticRegression, RidgeClassifier, PassiveAggressiveClassifier, Perceptron) , Ensemble models(RandomForest classifier, AdaBoostClassifier) and SVM model(LinearSVC). We use the accuracy score to measure the performance of the model (precision score, recall and confusion matrix are also calculated).

K-fold cross-validation is performed with each classifier to verify the consistency in the performance of the model using cross_validate module in sklearn. The number of folds is set to 10 and the mean and standard deviation of cross-validation scores are noted. The time taken for training each classifier is also measured using time.time() function

In [62]:

```
from sklearn.svm import LinearSVC
from sklearn.ensemble import AdaBoostClassifier
from sklearn.naive_bayes import MultinomialNB, BernoulliNB
from sklearn.linear_model import RidgeClassifier
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.linear_model import Perceptron
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier

from sklearn.pipeline import make_pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score,precision_score,recall_score
from sklearn.model_selection import cross_validate,KFold
import datetime
import time
import pandas as pd
#a list of classes
classifiers = [MultinomialNB(),BernoulliNB(),LogisticRegression(),LinearSVC(),AdaBoostClassifier(),RidgeClassifier(),Passive
clf_names = ['MultinomialNB()','BernoulliNB()','LogisticRegression()','LinearSVC()','AdaBoostClassifier()','RidgeClassifier(
data=[]

for v in ['cv','tf']:

    for gram in range(1,4):
        i=0

        for clf in classifiers:

            if(clf=='RandomForest Classifier'):#special case
                clf = RandomForestClassifier(random_state=0,n_jobs=-1,class_weight="balanced")

            before = datetime.datetime.now()
            before = before.strftime("%H:%M:%S")
            start = time.time()

            if(v=='cv'):
                vec = TfidfVectorizer(ngram_range=(1,gram))
            else:
                vec = CountVectorizer(ngram_range=(1,gram))

            model = make_pipeline(vec,clf)
            model.fit(x_train_copy.values.astype('U'),y_train_copy.values.astype('U'))##
            labels = model.predict(x_val_copy.values.astype('U'))
            ac = accuracy_score(y_val_copy.values.astype('U'),labels)
            kfold = KFold(n_splits=10,shuffle=False,random_state=None)
            results = cross_validate(model,x_train_copy.values.astype('U'),y_train_copy.values.astype('U'),cv=kfold)
            crossval_test_score_mean=results['test_score'].mean()
            crossval_train_score_mean=results['train_score'].mean()
            crossval_test_score_std=results['test_score'].std()
            crossval_train_score_std=results['train_score'].std()
```

Importing sklearn libraries and performing the Traning of algorithms.

Chapter 6: Conclusion

In the Major research project titled " Sentiment analysis of Nike-Dream Further Campaign using Python" by using the NLTK and various machine learning algorithms where we used the YouTube comments data using API and it can be concluded that the Campaign is a very successful and this can be verified using the positive Comments count vs the negative comments counts which are almost half of positive comments. This can be viewed from the Fig. 4.2 where the pie chart explains that the campaign is successful. This model is also verified using the accuracy got in the further below results from the various machine learning models. Using various Visualisation techniques like word cloud it explains the words which are used most frequently by customers in their comments on social media.

Youtube video used for the analysis.

After Implementation of the various Machine learning algorithms we print the accuracy of every algorithm for each of the N-grams In simple words, n-grams are all combinations of adjacent words that we can find in our text(comments). All combinations possible for the sentence "I like toys": unigrams- ('I', 'like', 'toys'), bigrams- ('I like', 'like toys'), trigrams- ('I like toys'). Both CV and Tfidf gives a parameter called ngram_range. The ngram_range as (1,1) denotes uni-gram, (1,2) a bigram, (1,3) a trigram and so on. We implement uni, bi, and trigrams with both CV and Tfidf and observe which of them gives more accuracy score.

	Vec_Gram	Classifier	Ac	crossval_train_score_mean	crossval_tes
0	cv_1	MultinomialNB()	0.632911	0.814866	
1	cv_1	BernoulliNB()	0.607595	0.725150	
2	cv_1	LogisticRegression()	0.620253	0.845083	
3	cv_1	LinearSVC()	0.721519	0.983759	
4	cv_1	AdaBoostClassifier()	0.645570	0.736237	
5	cv_1	RidgeClassifier()	0.721519	0.973295	
6	cv_1	PassiveAggressiveClassifier()	0.696203	0.986648	
7	cv_1	Perceptron()	0.708861	0.974153	
8	cv_1	RandomForest Classifier	0.708861	0.977278	
9	cv_2	MultinomialNB()	0.582278	0.896931	
10	cv_2	BernoulliNB()	0.443038	0.629733	
11	cv_2	LogisticRegression()	0.620253	0.897477	
12	cv_2	LinearSVC()	0.708861	0.996564	
13	cv_2	AdaBoostClassifier()	0.670886	0.739985	
14	cv_2	RidgeClassifier()	0.696203	0.994612	
15	cv_2	PassiveAggressiveClassifier()	0.746835	0.996486	
16	cv_2	Perceptron()	0.658228	0.996018	
17	cv_2	RandomForest Classifier	0.683544	0.976966	
18	cv_3	MultinomialNB()	0.569620	0.927930	
19	cv_3	BernoulliNB()	0.379747	0.586084	
20	cv_3	LogisticRegression()	0.620253	0.918560	
21	cv_3	LinearSVC()	0.683544	0.996564	
22	cv_3	AdaBoostClassifier()	0.683544	0.737409	
23	cv_3	RidgeClassifier()	0.683544	0.995705	
24	cv_3	PassiveAggressiveClassifier()	0.721519	0.996564	
25	cv_3	Perceptron()	0.696203	0.995784	
26	cv_3	RandomForest Classifier	0.683544	0.973998	
27	tf_1	MultinomialNB()	0.658228	0.859608	
28	tf_1	BernoulliNB()	0.607595	0.725150	
29	tf_1	LogisticRegression()	0.721519	0.946982	
30	tf_1	LinearSVC()	0.721519	0.989303	
31	tf_1	AdaBoostClassifier()	0.632911	0.735770	
32	tf_1	RidgeClassifier()	0.658228	0.954556	
33	tf_1	PassiveAggressiveClassifier()	0.683544	0.980479	
34	tf_1	Perceptron()	0.696203	0.965017	
35	tf_1	RandomForest Classifier	0.670886	0.977434	
36	tf_2	MultinomialNB()	0.645570	0.936129	

Output of Accuracy score of various algorithms

	Vec_Gram	Classifier	Ac	crossval_train_score_mean	crossval_tes
37	tf_2	BernoulliNB()	0.443038	0.629733	
38	tf_2	LogisticRegression()	0.696203	0.974233	
39	tf_2	LinearSVC()	0.708861	0.995159	
40	tf_2	AdaBoostClassifier()	0.607595	0.736004	
41	tf_2	RidgeClassifier()	0.696203	0.983369	
42	tf_2	PassiveAggressiveClassifier()	0.772152	0.996018	
43	tf_2	Perceptron()	0.658228	0.993753	
44	tf_2	RandomForest Classifier	0.696203	0.974857	
45	tf_3	MultinomialNB()	0.620253	0.951355	
46	tf_3	BernoulliNB()	0.379747	0.586084	
47	tf_3	LogisticRegression()	0.683544	0.973765	
48	tf_3	LinearSVC()	0.721519	0.996486	
49	tf_3	AdaBoostClassifier()	0.607595	0.736004	
50	tf_3	RidgeClassifier()	0.696203	0.980479	
51	tf_3	PassiveAggressiveClassifier()	0.772152	0.996252	
52	tf_3	Perceptron()	0.670886	0.994690	
53	tf_3	RandomForest Classifier	0.683544	0.973217	

Output of Accuracy score of various algorithms

In the image, 'Ac' represents the accuracy score and 'cv_1' means CountVectorizer with unigrams. In terms of cross-validation mean score and standard deviation, LinearSVC seems better. It is worth noting that our dataset is imbalanced- with more positive and negative comments than neutral comments. So, in this case, the accuracy scores may lead us to wrong conclusions. To avoid that, we also used Random Forest Classifier with balanced class weights. Below are the models which provided the accuracy greater than 70%.

```
d[d.Ac>0.70]
```

Out[64]:

	Vec_Gram	Classifier	Ac	crossval_train_score_mean	crossval_tes
3	cv_1	LinearSVC()	0.721519	0.983759	
5	cv_1	RidgeClassifier()	0.721519	0.973295	
7	cv_1	Perceptron()	0.708861	0.974153	
8	cv_1	RandomForest Classifier	0.708861	0.977278	
12	cv_2	LinearSVC()	0.708861	0.996564	
15	cv_2	PassiveAggressiveClassifier()	0.746835	0.996486	
24	cv_3	PassiveAggressiveClassifier()	0.721519	0.996564	
29	tf_1	LogisticRegression()	0.721519	0.946982	
30	tf_1	LinearSVC()	0.721519	0.989303	
39	tf_2	LinearSVC()	0.708861	0.995159	
42	tf_2	PassiveAggressiveClassifier()	0.772152	0.996018	
48	tf_3	LinearSVC()	0.721519	0.996486	
51	tf_3	PassiveAggressiveClassifier()	0.772152	0.996252	

Output of Accuracy score of algorithms having accuracy > 70%

References

1. https://medium.com/datadriveninvestor/python-data-science-getting-started-tutorial-nltk-2d8842fedfdd
2. https://medium.com/analytics-vidhya/pythons-natural-language-tool-kit-nltk-tutorial-part-1-645688219a91
3. https://towardsdatascience.com/data-preprocessing-concepts-fa946d11c825
4. https://www.microstrategy.com/us/resources/introductory-guides/data-visualization-what-it-is-and-why-we-use-it
5. https://www.analyticsvidhya.com/blog/2018/02/natural-language-processing-for-beginners-using-textblob/
6. https://towardsdatascience.com/creating-word-clouds-with-python-f2077c8de5cc
7. https://www.aclweb.org/anthology/W11-0705.pdf
8. https://towardsdatascience.com/data-wrangling-with-pandas-5b0be151df4e
9. https://www.datacamp.com/community/news/data-mining-in-a-nutshell-chq5vml8xn
10. https://www.datacamp.com/courses/intro-to-python-for-data-science
11. https://www.datacamp.com/courses/intermediate-python-for-data-science
12. https://www.datacamp.com/courses/intermediate-python-for-data-science
13. https://www.datacamp.com/courses/cleaning-data-in-python
14. https://www.datacamp.com/courses/pandas-foundations
15. https://www.guru99.com/etl-extract-load-process.html
16. https://www.geeksforgeeks.org/etl-process-in-data-warehouse/
17. https://www.geeksforgeeks.org/data-cleansing-introduction/
18. https://ahrefs.com/blog/google-advanced-search-operators/
19. https://ahrefs.com/blog/google-advanced-search-operators/