

Chapter 1 Sentiment Analysis of Amazon Reviews
A Project Report for Industrial Internship

Submitted By
Sneha Ghosh

In the partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

Electronics and Communication Engineering

OF

**B. P. PODDAR INSTITUTE OF MANAGEMENT AND
TECHNOLOGY, KOLKATA.**



Under the Guidance of
MAHENDRA DATTA

Project Carried Out
At



Ardent Computech Pvt. Ltd. (An ISO 9001:2015 Company)
SDF Building, Module 132, Ground Floor, Sector V, Salt Lake, Kolkata
700091

In Association With



1. Title of the Project: Sentiment Analysis of Amazon Reviews
2. Project Member: Satyapriya Ghosh, Suvojit Das, Sneha Ghosh, Rupam Dey.
3. Name and Address of the Guide: Mahendra Datta

Ardent Computech Pvt. Ltd. (An ISO 9001:2015 Company)

Module No.-132, SDF Building, Sector V, Salt Lake, Kolkata Pin - 700091.

4. Educational Qualification of the Guide: B. Tech, MAKAUT.
5. Working / Training experience of the Guide:

PROJECT VERSION CONTROL HISTORY

| VERSION | PRIMARY AUTHOR | DESCRIPTION OF VERSION | DATE COMPLETED |
|---------|------------------|------------------------|----------------|
| Final | Suvojit Das | Project Report | |
| Final | Sneha Ghosh | Project Report | |
| Final | Rupam Dey | Project Report | |
| Final | Satyapriya Ghosh | Project Report | |

Date: -

Name of the Student: Sneha Ghosh

Signature of Approver

Signature of the Student

Date: -

MR. MAHENDRA DATTA

Project Proposal Evaluator

Approved

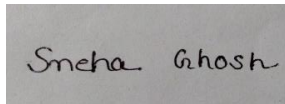
Not Approved

DECLARATION

We hereby declare that the project work being presented in the project proposal entitled “**Sentiment Analysis of Amazon Reviews**” in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** at **ARDENT COMPUTECH PVT. LTD, SALT LAKE, KOLKATA, WEST BENGAL**, is an authentic work carried out under the guidance of **MR. MAHENDRA DATTA**. The matter embodied in this project work has not been submitted elsewhere for the award of any degree of our knowledge and belief.

Date: -

Name of the Student: - Sneha Ghosh



Signature of the Student



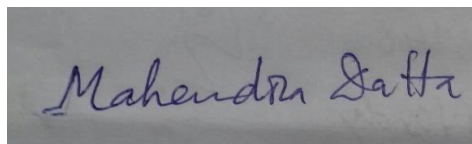
Ardent Computech Pvt. Ltd (An ISO 9001:2015 Certified)

SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal 700091

CERTIFICATE

This is to certify that this proposal of the minor project entitled “**Sentiment Analysis of Amazon Reviews**” is a record of bona fide work, carried out by **SNEHA GHOSH** under my guidance at **ARDENT COMPUTECH PVT LTD**. In my opinion, the report in its present form is in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** and as per regulations of the **ARDENT**. To the best of my knowledge, the results embodied in this report, are original in nature and worthy of incorporation in the present version of the report.

Guide / Supervisor



MR. MAHENDRA DATTA

Project Engineer

Ardent Computech Pvt. Ltd (An ISO 9001:2015 Certified)

SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal 700091

ACKNOWLEDGEMENT

Success of any project depends largely on the encouragement and guidelines of many others. I take this sincere opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project work.

I would like to show our greatest appreciation to ***Mr. MAHENDRA DATTA***, Project Engineer at Ardent, Kolkata. I always feel motivated and encouraged every time by his valuable advice and constant inspiration; without his encouragement and guidance this project would not have materialized.

Words are inadequate in offering our thanks to the other trainees, project assistants and other members at Ardent Computech Pvt. Ltd. for their encouragement and cooperation in carrying out this project work. The guidance and support received from all the members and who are contributing to this project, was vital for the success of this project.

CONTENTS

- Overview
- History of Python
- Environment Setup
- Basic Syntax
- Variable Types
- Functions
- Modules
- Packages
- Artificial Intelligence
 - Machine Learning
- Machine Learning
 - Supervised and Unsupervised Learning
 - NumPy
 - Scikit-learn
 - Pandas
 - Regression Analysis
 - Matplotlib
 - Clustering
 - Wordcloud
 - nltk
- SENTIMENT ANALYSIS OF AMAZON REVIEWS.
 1. Introduction
 2. Problem Statement
 3. Advantages & Disadvantages
 4. Future Scope
 5. Actual Code for Sentiment Analysis of Amazon Reviews
- Conclusion
- Future Scope for this Project
- BIBLIOGRAPHY

OVERVIEW

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and has fewer syntactical constructions than other languages.

Python is interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to Perl and PHP.

Python is Interactive: You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python is Object-Oriented: Python supports the Object-Oriented style or technique of programming that encapsulates code within objects.

Python is a Beginner's Language: Python is a great language for beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

HISTORY OF PYTHON

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Small Talk, UNIX shell, and other scripting languages. Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL). Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

FEATURES OF PYTHON

Easy-to-learn: Python has few keywords, a simple structure, and clearly defined syntax. This allows a student to pick up the language quickly.

Easy-to-Read: Python code is more clearly defined and visible to the eyes.

Easy-to-Maintain: Python's source code is fairly easy-to-maintain.

A broad standard library: Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

Interactive Mode: Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

Portable: Python can run on the wide variety of hardware platforms and has the same interface on all platforms.

Extendable: You can add low-level modules to the python interpreter. These modules enables programmers to add to or customize their tools to be more efficient.

Databases: Python provides interfaces to all major commercial databases.

GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

Scalable: Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below:

- It support functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collections.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and JAVA.

ENVIRONMENT SETUP

Open a terminal window and type "python" to find out if it is already installed and which version is installed.

- UNIX (Solaris, Linux, FreeBSD, AIX, HP/UX, SunOS, IRIX, etc.)
- Win 9x/NT/2000
- Macintosh (Intel, PPC, 68K)
- OS/2
- DOS (multiple versions)
- PalmOS
- Nokia mobile phones
- Windows CE
- Acorn/RISC OS

BASIC SYNTAX OF PYTHON PROGRAM

Type the following text at the Python prompt and press the Enter –

```
>>> print "Hello, Python!"
```

*If you are running new version of Python, then you would need to use print statement with parenthesis as in **print ('Hello, Python!');***

However, in Python version 2.4.3, this produces the following result –

Hello, Python!

Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, \$, and % within identifiers. Python is a case sensitive programming language.

Python Keywords

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

And, exec, not
Assert, finally, or
Break, for, pass
Class, from, print
continue, global, raise
def, if, return
del, import, try
elif, in, while
else, is, with
except, lambda, yield

Lines & Indentation

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.

The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example –

```
if True:
    print "True"
else:
    print "False"
```

Command Line Arguments

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -h –

```
$ python-h
usage: python [option]...[-c cmd|-m mod | file |-][arg]...
```

Options and arguments (and corresponding environment variables):

- c cmd: program passed in as string(terminates option list)
- d : debug output from parser (also PYTHONDEBUG=x)
- E : ignore environment variables (such as PYTHONPATH)
- h : print this help message and exit [etc.]

VARIABLE TYPES

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Assigning Values to Variables

Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables.

```
counter=10      # An integer assignment
weight=10.60    # A floating point
name="Ardent"   # A string
```

Multiple Assignment

Python allows you to assign a single value to several variables simultaneously. For example –

```
a = b = c = 1
a,b,c = 1,2,"hello"
```

Standard Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has five standard data types –

- ☐ String
- ☐ List
- ☐ Tuple
- ☐ Dictionary
- ☐ Number

Data Type Conversion

Sometimes, you may need to perform conversions between the built-in types. To convert between types, you simply use the type name as a function.

There are several built-in functions to perform conversion from one data type to another.

| <u>SL.NO.</u> | <u>Function And Description</u> |
|---------------|---|
| <u>1.</u> | int(x [,base]) Converts x to an integer, base specifies the base if % is a string. |
| <u>2.</u> | long(x [,base]) Converts x to a long integer, base specifies the base if % is a string. |
| <u>3.</u> | float(x) Converts x to a floating-point number, |
| <u>4.</u> | complex(real [,imag]) Creates a complex number. |
| <u>5.</u> | str(x) Converts object x to a string representation. |
| <u>6.</u> | repr(x) Converts object x to an expression sting. |
| <u>7.</u> | eval(str) Evaluates a string and returns an object. |
| <u>8.</u> | tuple(s) Converts s to a tuple. |
| <u>9.</u> | list(s) Converts s to a list. |

FUNCTIONS

Defining a Function

- `def function name (parameters):`
`"function docstring"`
`function suite`
`return [expression]`

Pass by reference vs Pass by value

All parameters (arguments) in the Python language are passed by reference. It means if you change what a parameter refers to within a function, the change also reflects back in the calling function. For example –

Function definition is here

```
def change me(mylist):  
    "This changes a passed list into this function"  
    mylist.append([1,2,3,4]);  
    print"Values inside the function: ",mylist  
    return
```

Now you can call changeme function

```
mylist=[10,20,30];  
change me(mylist);  
print" Values outside the function: ",mylist
```

Here, we are maintaining reference of the passed object and appending values in the same object. So, this would produce the following result –

Values inside the function: [10, 20, 30, [1, 2, 3, 4]]
Values outside the function: [10, 20, 30, [1, 2, 3, 4]]

Global vs. Local variables

Variables that are defined inside a function body have a local scope, and those defined outside have a global scope . For Example-

```
total=0;          # This is global variable.
```

```
# Function definition is here
```

```
def sum( arg1, arg2 ):
```

```
# Add both the parameters and return them."
```

```
total= arg1 + arg2;    # Here total is local variable.  
print"Inside the function local total: ", total  
return total;
```

```
# Now you can call sum function
```

```
sum(10,20);  
Print"Outside the function global total: ", total
```

When the above code is executed, it produces the following result –

```
Inside the function local total: 30  
Outside the function global total: 0
```


MODULES

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference.

The Python code for a module named *aname* normally resides in a file named *aname.py*. Here's an example of a simple module, *support.py*

```
def print_func( par ):  
    print "Hello : ", par  
    return
```

The *import* Statement

You can use any Python source file as a module by executing an *import* statement in some other Python source file. The *import* has the following syntax

—

```
Import module1 [, module2 [... moduleN]
```

PACKAGES

A package is a hierarchical file directory structure that defines a single Python application environment that consists of modules and sub packages and sub-subpackages, and so on.

Consider a file *Pots.py* available in *Phone* directory. This file has following line of source code –

```
def Pots ():  
    print "I'm Pots Phone"
```

Similar way, we have another two files having different functions with the same name as above –

- ☐ *Phone/Isdn.py* file having function *Isdn ()*
- ☐ *Phone/G3.py* file having function *G3 ()*

Now, create one more file *__init__.py* in *Phone* directory –

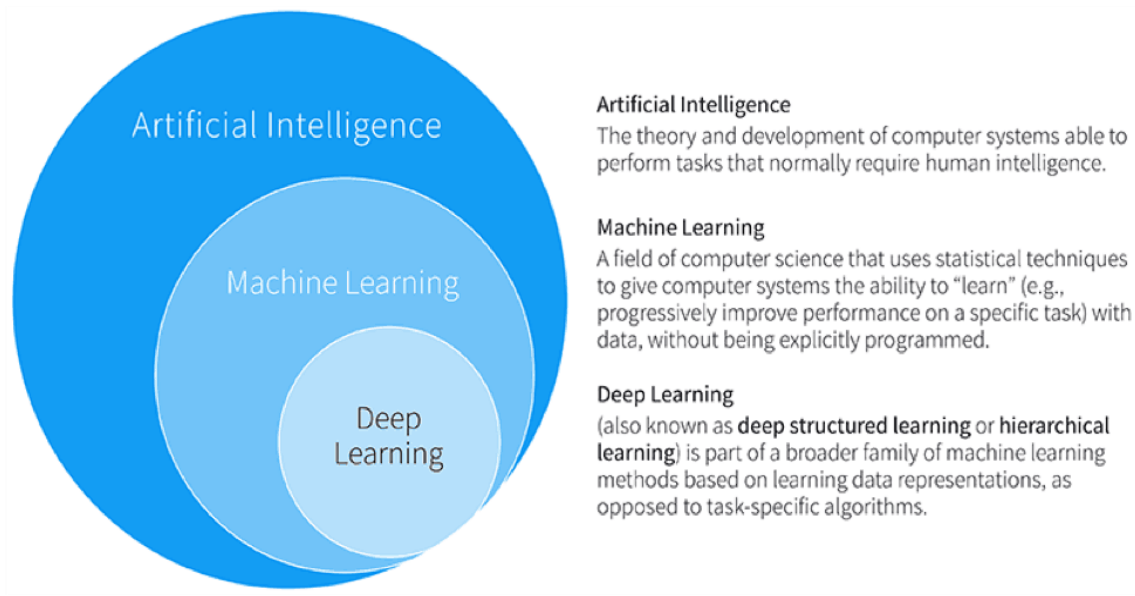
- ☐ *Phone/__init__.py*

To make all of your functions available when you've imported *Phone*, you need to put explicit import statements in *__init__.py* as follows –

```
from Pots import Pots  
from Isdn import Isdn  
from G3 import
```

ARTIFICIAL INTELLIGENCE

Introduction



According to the father of Artificial Intelligence, John McCarthy, it is “*The science and engineering of making intelligent machines, especially intelligent computer programs*”.

Artificial Intelligence is a way of **making a computer, a computer-controlled robot, or a software think intelligently**, in the similar manner the intelligent humans think.

AI is accomplished by studying how human brain thinks, and how humans learn, decide, and work while trying to solve a problem, and then using the outcomes of this study as a basis of developing intelligent software and systems.

The development of AI started with the intention of creating similar intelligence in machines that we find and regard high in humans.

Goals of AI

To Create Expert Systems – The systems which exhibit intelligent behaviour, learn, demonstrate, explain, and advice its users.

To Implement Human Intelligence in Machines – Creating systems that understand, think, learn, and behave like humans.

Applications of AI

AI has been dominant in various fields such as :-

Gaming – AI plays crucial role in strategic games such as chess, poker, tic-tac-toe, etc., where machine can think of large number of possible positions based on heuristic knowledge.

Natural Language Processing – It is possible to interact with the computer that understands natural language spoken by humans.

Expert Systems – There are some applications which integrate machine, software, and special information to impart reasoning and advising. They provide explanation and advice to the users.

Vision Systems – These systems understand, interpret, and comprehend visual input on the computer.

For example: A spying aeroplane takes photographs, which are used to figure out spatial information

Or map of the areas.

Doctors use clinical expert system to diagnose the patient.

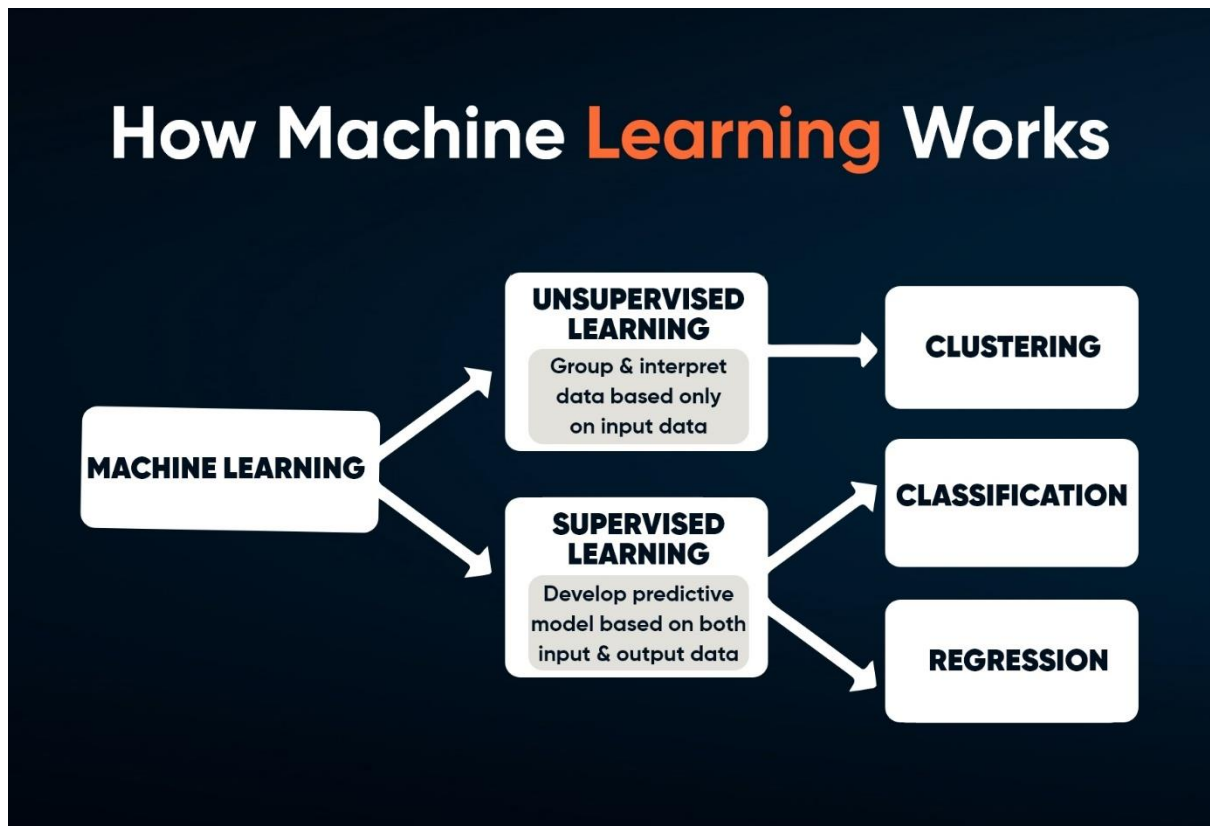
Police use computer software that can recognize the face of criminal with the stored portrait made by forensic artist.

Speech Recognition – Some intelligent systems are capable of hearing and comprehending the language in terms of sentences and their meanings while a human talks to it. It can handle different accents, slang words, noise in the background, change in human's noise due to cold, etc.

Handwriting Recognition – The handwriting recognition software reads the text written on paper by a pen or on screen by a stylus. It can recognize the shapes of the letters and convert it into editable text.

Intelligent Robots – Robots are able to perform the tasks given by a human. They have sensors to detect physical data from the real world such as light, heat, temperature, movement, sound, bump, and pressure. They have efficient processors, multiple sensors and huge memory, to exhibit intelligence. In addition, they are capable of learning from their mistakes and they can adapt to the new environment.

MACHINE LEARNING



Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed.

Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data.

INTRODUCTION TO MACHINE LEARNING

Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed.

Arthur Samuel, an American pioneer in the field of computer gaming and artificial intelligence, coined the term "Machine Learning" in 1959 while at IBM. Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data

Machine learning tasks are typically classified into two broad categories, depending on whether there is a learning "signal" or "feedback" available to a learning system: -

SUPERVISED LEARNING

Supervised learning is the machine learning task of inferring a function from *labelled training data*.^[1] The training data consist of a set of *training examples*. In supervised learning, each example is a *pair* consisting of an input object (typically a vector) and a desired output value.

A supervised learning algorithm analyses the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

UNSUPERVISED LEARNING

Unsupervised learning is the machine learning task of inferring a function to describe hidden structure from "unlabelled" data (a classification or categorization is not included in the observations). Since the examples given to the learner are unlabelled, there is no evaluation of the accuracy of the structure that is output by the relevant algorithm—which is one way of distinguishing unsupervised learning from supervised learning and reinforcement learning.

A central case of unsupervised learning is the problem of density estimation in statistics, though unsupervised learning encompasses many other problems (and solutions) involving summarizing and explaining key features of the data.

NUMPY

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin.

NumPy targets the C Python reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars.

NUMPY ARRAY

NumPy's main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy dimensions are called *axes*. The number of axes is *rank*.

For example, the coordinates of a point in 3D space [1, 2, 1] is an array of rank 1, because it has one axis. That axis has a length of 3. In the example pictured below, the array has rank 2 (it is 2-dimensional). The first dimension (axis) has a length of 2, the second dimension has a length of 3.

```
[[1., 0., 0.],  
 [ 0., 1., 2.]]
```

NumPy's array class is called *ndarray*. It is also known by the alias.

SLICING NUMPY ARRAY

Import numpy as np

```
a = np.array ([[1, 2, 3],[3,4,5],[4,5,6]])
```

```
print 'Our array is:'
```

```
Print a
```

```
print '\n'
```

```
print 'The items in the second column are:'
```

```
print a[...,1]
```

```
print '\n'
```

```
print 'The items in the second row are:'
```

```
print a[1...]
```

```
print '\n'
```

```
print 'The items columns 1 onwards are:'
```

```
print a [...,1:]
```

OUTPUT

Our array is:

```
[[1 2 3]
```

```
[3 4 5]
```

```
[4 5 6]]
```

The items in the second column are:

```
[2 4 5]
```

The items in the second row are:

```
[3 4 5]
```

The items column 1 onwards are:

```
[[2 3]
```

```
[4 5]
```

```
[5 6]]
```


SCIKIT-LEARN

Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

The scikit-learn project started as scikit-learn, a Google Summer of Code project by David Cournapeau. Its name stems from the notion that it is a "SciKit" (SciPy Toolkit), a separately-developed and distributed third-party extension to SciPy.[4] The original codebase was later rewritten by other developers. In 2010 Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort and Vincent Michel, all from INRIA took leadership of the project and made the first public release on February the 1st 2010[5]. Of the various scikits, scikit-learn as well as scikit-image were described as "well-maintained and popular" in November 2012.

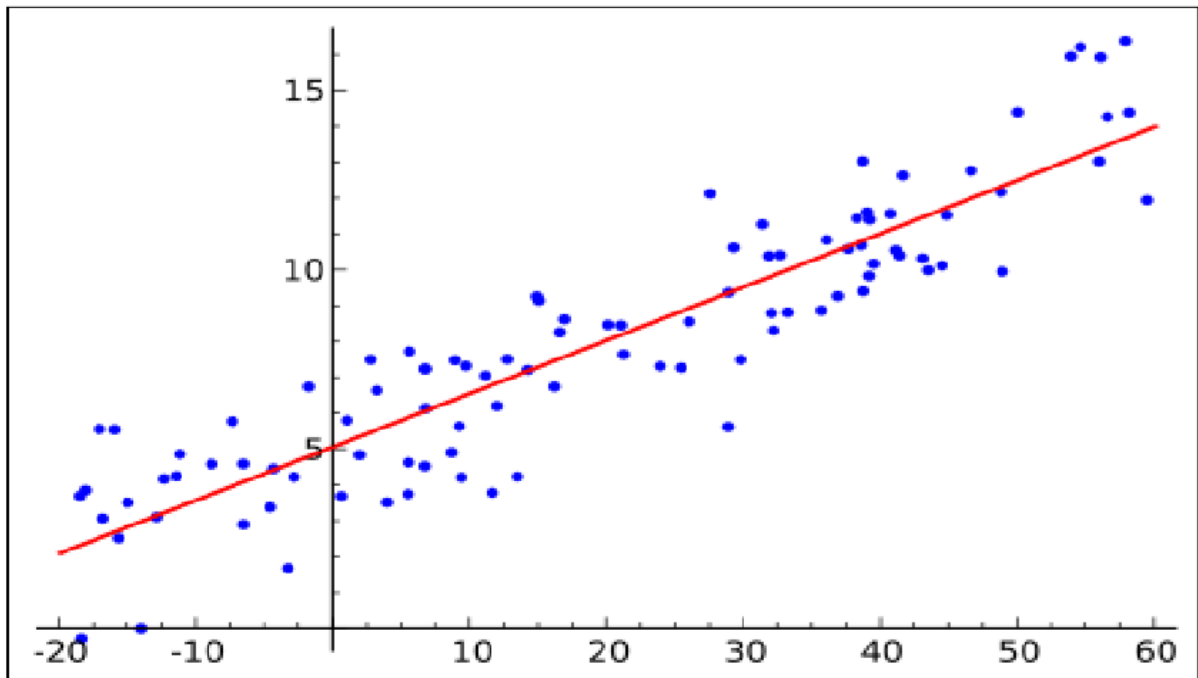
PANDAS

In computer programming, **pandas** is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. "Panel data", an econometrics term for multidimensional, structured data sets.

LIBRARY FEATURES

- Data Frame object for data manipulation with integrated indexing.
- Tools for reading and writing data between in-memory data structures and different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of data sets.
- Label-based slicing, fancy indexing, and sub setting of large data sets.
- Data structure column insertion and deletion.
- Group by engine allowing split-apply-combine operations on data sets.
- Data set merging and joining.
- Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure.
- Time series-functionality: Date range generation.

REGRESSION ANALYSIS



In statistical modelling, **regression analysis** is a set of statistical processes for estimating the relationships among variables. It includes many techniques for modelling and analysing several variables, when the focus is on the relationship between a dependent variable and one or more independent variables (or 'predictors'). More specifically, regression analysis helps one understand how the typical value of the dependent variable (or 'criterion variable') changes when any one of the independent variables is varied, while the other independent variables are held fixed.

Regression analysis is widely used for prediction and forecasting, where its use has substantial overlap with the field of machine learning. Regression analysis is also used to understand which among the independent variables are related to the dependent variable, and to explore the forms of these relationships. In restricted circumstances, regression analysis can be used to infer casual relationships between the independent and dependent variables. However, this can lead to illusions or false relationships, so caution is advisable

LINEAR REGRESSION

Linear regression is a linear approach for modelling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted X . The case of one explanatory variable is called *simple linear regression*. For more than one explanatory variable, the process is called *multiple linear regression*.

In linear regression, the relationships are modelled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called linear models.

LOGISTIC REGRESSION

Logistic regression, or logit regression, or logit model ^[1] is a regression model where the dependent variable (DV) is categorical. This article covers the case of a binary dependent variable—that is, where the output can take only two values, "0" and "1", which represent outcomes such as pass/fail, win/lose, alive/dead or healthy/sick. Cases where the dependent variable has more than two outcome categories may be analysed in multinomial logistic regression, or, if the multiple categories are ordered, in ordinal logistic regression. In the terminology of economics, logistic regression is an example of a qualitative response/discrete choice model.

POLYNOMIAL REGRESSION

Polynomial regression is a form of regression analysis in which the relationship between the independent variable x and the dependent variable y is modelled as an n^{th} degree polynomial in x .

Polynomial regression fits a nonlinear relationship between the value of x and the corresponding conditional mean of y , denoted $E(y | x)$, and has been used to describe nonlinear phenomena such as the growth rate of tissues, the distribution of carbon isotopes in lake sediments, and the progression of disease epidemics.

Although *polynomial regression* fits a nonlinear model to the data, as a statistical estimation problem it is linear, in the sense that the regression function $E(y | x)$ is linear in the unknown parameters that are estimated from the data.

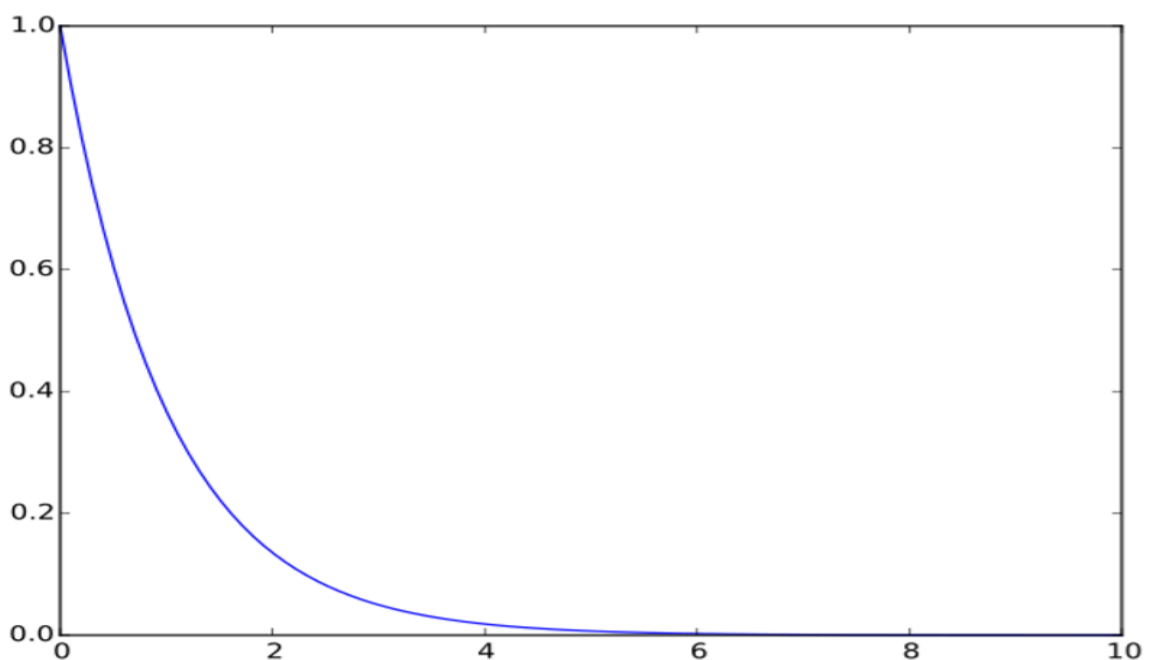
MATPLOTLIB

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of matplotlib.

EXAMPLE

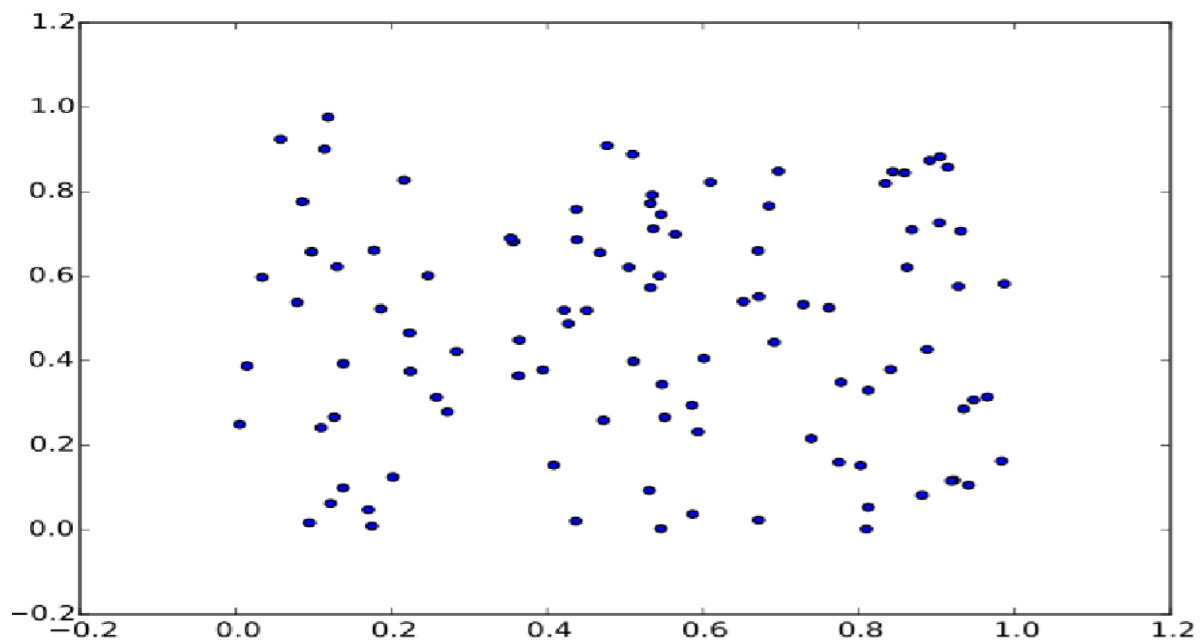
➤ LINE PLOT

```
>>>importmatplotlib.pyplotasplt
>>>importnumpyasnp
>>>a=np.linspace(0,10,100)
>>>b=np.exp(-a)
>>>plt.plot(a,b)
>>>plt.show()
```



➤ SCATTER PLOT

```
>>>importmatplotlib.pyplotasplt
>>>fromnumpy.randomimportrand
>>>a=rand(100)
>>>b=rand(100)
>>>plt.scatter(a,b)
>>>plt.show()
```



CLUSTERING

Cluster analysis or **clustering** is the task of grouping a set of objects in such a way that objects in the same group (called a **cluster**) are more similar (in some sense or another) to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics.

Cluster analysis itself is not one specific algorithm, but the general task to be solved. It can be achieved by various algorithms that differ significantly in their notion of what constitutes a cluster and how to efficiently find them. Popular notions of clusters include groups with small distances among the cluster members, dense areas of the data space, intervals or particular statistical distributions. Clustering can therefore be formulated as a multi-objective optimization problem.

The appropriate clustering algorithm and parameter settings (including values such as the distance function to use, a density threshold or the number of expected clusters) depend on the individual data set and intended use of the results. Cluster analysis as such is not an automatic task, but an iterative process of knowledge discovery or interactive multi-objective optimization that involves trial and failure. It is often necessary to modify data pre-processing and model parameters until the result achieves the desired properties.

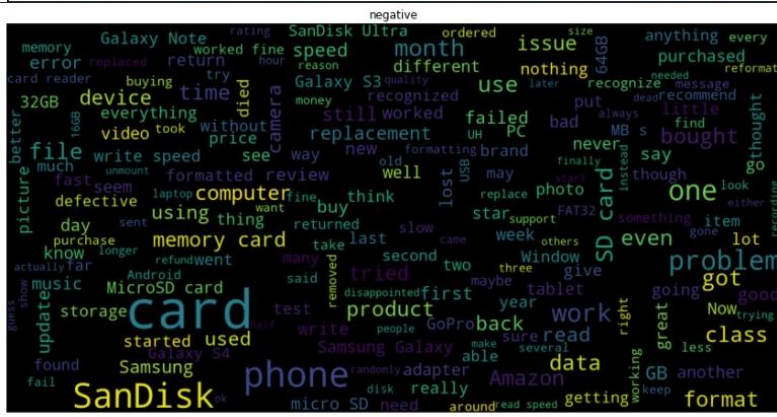
WORDCLOUD

Word Cloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance. Significant textual data points can be highlighted using a word cloud. Word clouds are widely used for analysing data from social network websites. For generating word cloud in Python, modules needed are matplotlib, pandas and wordcloud.

The below code will display the wordcloud :

```
import pandas as pd
import matplotlib.pyplot as plt
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from wordcloud import WordCloud
f=pd.read_csv("amazon_reviews.csv")
df.loc[df['overall']<=3,'overall']=0
df.loc[df['overall']>3,'overall']=1
stp_words=stopwords.words('english')
def clean_review(review):
    cleanreview=" ".join(word for word in review.split() if word not in stp_words)
    return cleanreview
df['reviewText']=df['reviewText'].apply(clean_review)
#---create the wordcloud with overall 0 or negative.---
consolidated=' '.join(word for word in df['reviewText'][df['overall']==0].astype(str))
wordCloud=WordCloud(width=1600,height=800,random_state=21,max_font_size=110)
plt.figure(figsize=(15,10))
plt.imshow(wordCloud.generate(consolidated),interpolation='bilinear')
plt.axis('off')
plt.title('negative')
plt.show()
#---create the wordcloud with overall 1 or positive .---
consolidated=' '.join(word for word in df['reviewText'][df['overall']==1].astype(str))
wordCloud=WordCloud(width=1600,height=800,random_state=21,max_font_size=110)
plt.figure(figsize=(15,10))
```


plt.show()



NLTK

Natural Language Processing (NLP) is a field that focuses on making natural human language usable by computer programs. **NLTK**, Natural Language Toolkit, is a Python package that you can use for NLP.

A lot of the data that you could be analyzing is unstructured data and contains human-readable text. Before you can analyze that data programmatically, you first need to preprocess it. In this tutorial, you'll take your first look at the kinds of **text preprocessing** tasks you can do with NLTK so that you'll be ready to apply them in future projects. You'll also see how to do some basic **text analysis** and create **visualizations**.

Tokenizing

By **tokenizing**, you can conveniently split up text by word or by sentence. This will allow you to work with smaller pieces of text that are still relatively coherent and meaningful even outside of the context of the rest of the text. It's your first step in turning unstructured data into structured data, which is easier to analyze.

Filtering Stop Words

Stop words are words that you want to ignore, so you filter them out of your text when you're processing it. Very common words like 'in', 'is', and 'an' are often used as stop words since they don't add a lot of meaning to a text in and of themselves.

Here's how to import the relevant parts of NLTK in order to filter out stop words:

```
>>> nltk.download("stopwords")
>>> from nltk.corpus import stopwords
>>> from nltk.tokenize import word_tokenize
```

Here's a [quote from Worf](#) that you can filter:

```
>>> worf_quote = "Sir, I protest. I am not a merry man!"
```

Now tokenize worf_quote by word and store the resulting list in words_in_quote

```
>>> words_in_quote = word_tokenize(worf_quote)
>>> words_in_quote
['Sir', ',', 'protest', '.', 'merry', 'man', '!']
```

You have a list of the words in `words_in_quote`, so the next step is to create a set of stop words to filter `words_in_quote`. For this example, you'll need to focus on stop words in "english":

```
>>> stop_words = set(stopwords.words("english"))
```

Next, create an empty list to hold the words that make it past the filter:

```
>>> filtered_list = []
```

You created an empty list, `filtered_list`, to hold all the words in `words_in_quote` that aren't stop words. Now you can use `stop_words` to filter `words_in_quote`:

```
>>> for word in words_in_quote:
...     if word.casefold() not in stop_words:
...         filtered_list.append(word)
```

You iterated over `words_in_quote` with a for loop and added all the words that weren't stop words to `filtered_list`. You used `casefold()` on `word` so you could ignore whether the letters in `word` were uppercase or lowercase. This is worth doing because `stopwords.words('english')` includes only lowercase versions of stop words.

Stemming

Stemming is a text processing task in which you reduce words to their root, which is the core part of a word. For example, the words "helping" and "helper" share the root "help." Stemming allows you to zero in on the basic meaning of a word rather than all the details of how it's being used. NLTK has more than one stemmer, but you'll be using the Porter Stemmer.

Tagging Parts of Speech

Part of speech is a grammatical term that deals with the roles words play when you use them together in sentences. Tagging parts of speech, or **POS tagging**, is the task of labelling the words in your text according to their part of speech.

Lemmatizing

Now that you're up to speed on parts of speech, you can circle back to lemmatizing. Like stemming, **lemmatizing** reduces words to their core meaning, but it will give you a complete English word that makes sense on its own instead of just a fragment of a word like 'discoveri'.

Chunking

While tokenizing allows you to identify words and sentences, **chunking** allows you to identify **phrases**.

ALGORITHM

- Data Collection
- Data Formatting
- Model Selection
- Training
- Testing

Data Collection: We have collected data sets of weather from online website. We have downloaded the .csv files in which information was present.

Data Formatting: The collected data is formatted into suitable data sets. We check the collinearity with mean temperature. The data sets which have collinearity nearer to 1.0 has been selected.

Model Selection: We have selected different models to minimize the error of the predicted value. The different models used are Linear Regression Linear Model, Ridge Linear model, Lasso Linear Model and Bayesian Ridge Linear Model.

Training: The data set was divided such that x_train is used to train the model with corresponding x_test values and some y_train kept reserved for testing.

Testing: The model was tested with y_train and stored in y_predict. Both y_train and y_predict was compared.

Chapter 2

AMAZON REVIEW SENTIMENT ANALYSIS BY USING MACHINE LEARNING

INTRODUCTION

Sentiment analysis is a branch of natural language processing (NLP) that involves determining the sentiment or opinion expressed in a piece of text. In this case, we'll focus on analyzing the sentiment of Amazon reviews using machine learning techniques.

To perform sentiment analysis, we'll need a dataset of Amazon reviews labeled with their corresponding sentiments (positive, negative, or neutral). This dataset can be obtained by manually annotating a set of reviews or by using pre-labeled datasets available online. Once we have the dataset, we can proceed with the following steps:

1. Data Preprocessing:

- Clean the text data by removing any irrelevant information such as HTML tags, punctuation, and special characters.
- Convert the text to lowercase to ensure consistent processing.
- Tokenize the text by splitting it into individual words or phrases.
- Remove stop words (common words that do not carry much meaning) such as "the," "is," "and," etc.

2. Analysis of dataset:

- Here we counts how many positive and negative sentiments are there.
- To have a better picture of the importance of the words we have created word cloud of all the words with sentiment = 0 as well as sentiment = 1

3. Feature Extraction:

- a. Represent the text using numerical features that can be understood by machine learning algorithms.
- b. One common approach is to use the Bag-of-Words model, where each review is represented as a vector of word frequencies.

- c. Another approach is to use word embeddings like Word2Vec or GloVe, which capture semantic relationships between words.

4. Model Training:

- a. Split the labeled dataset into training and testing sets.
- b. Train a machine learning model such as logistic regression, support vector machines, or a deep learning model like recurrent neural networks (RNNs) or transformers.
- c. Train the model on the labeled training data, using the extracted features and corresponding sentiment labels.

5. Model Evaluation:

- a. Evaluate the trained model on the testing data to measure its performance.
- b. Common evaluation metrics for sentiment analysis include accuracy, precision, recall, and F1-score.
- c. Make adjustments to the model or feature extraction techniques if necessary to improve performance.

6. Prediction:

- a. Once the model is trained and evaluated, it can be used to predict the sentiment of new, unlabeled Amazon reviews.
- b. Apply the same preprocessing steps to the new reviews as done during training.
- c. Use the trained model to predict the sentiment of the new reviews based on the extracted features.

It's worth noting that the effectiveness of sentiment analysis can vary depending on the quality and representativeness of the labelled dataset, as well as the chosen machine learning model and feature extraction techniques. Continuous improvement and fine-tuning may be necessary to achieve higher accuracy and reliability in sentiment classification.

About Dataset

Context

Access to safe drinking water is essential to health, a basic human right, and a component of effective policy for health protection. This is important as a health and development issue at a national, regional, and local level. In some regions, it has been shown that investments in water supply and sanitation can yield a net economic benefit, since the reductions in adverse health effects and health care costs outweigh the costs of undertaking the interventions.

PROBLEM STATEMENT

Sentiment Analysis of Amazon Reviews

Content

The amazon_reviews.csv file contains reviews and ratings for 4914 different users or consumers data. It contains Amazon Product Data includes product categories and various metadata. The product with the most comments in the electronics category has user ratings and comments. In this way, we expect you to perform sentiment analysis with your specific methods.

| Unnamed: 0 | reviewerName | overall | reviewText | reviewTime | day_diff | helpful_yes | helpful_no | total_vote | score_pos_neg_diff | score_average_rating | wilson_low |
|------------|--------------|----------|------------|---|------------|-------------|------------|------------|--------------------|----------------------|------------|
| 0 | 0 | NaN | 4 | No issues. | 23-07-2014 | 138 | 0 | 0 | 0 | 0 | 0.0 |
| 1 | 1 | Omie | 5 | Purchased this for my device, it worked as adv... | 25-10-2013 | 409 | 0 | 0 | 0 | 0 | 0.0 |
| 2 | 2 | 1K3 | 4 | it works as expected. I should have sprung for... | 23-12-2012 | 715 | 0 | 0 | 0 | 0 | 0.0 |
| 3 | 3 | 1m2 | 5 | This think has worked out great.Had a diff. br... | 21-11-2013 | 382 | 0 | 0 | 0 | 0 | 0.0 |
| 4 | 4 | 2&1/2Men | 5 | Bought it with Retail Packaging, arrived legit... | 13-07-2013 | 513 | 0 | 0 | 0 | 0 | 0.0 |

The different columns are:

1. **reviewerName** : It contains the name of the users.
2. **overall**: It contains the rating given to a product out of 5.
3. **reviewText**: It contains the sentiments of user about a product either it is good or bad.
4. **reviewTime**: It contains the date of the given review.
5. **day_diff**: It gives the day difference between different reviews.
6. **helpful_yes**: It contains the value which is how many number of times a review helped another user either to buy the product or not.
7. **helpful_no**: It contains the values for the reviews which is not helpful for other users.
8. **total_vote**: It counts the total number of votes of a review either helpful or not helpful.
9. **score_pos_neg_diff**
10. **score_average_rating**
11. **wilson_lower_bound**

ADVANTAGES AND DISADVANTAGES

Advantages :

1. **Scalability:** Machine learning algorithms can process and analyze large amounts of data quickly and efficiently. This is particularly useful for analyzing a vast number of Amazon reviews, as there can be millions of reviews for popular products.
2. **Accuracy:** With proper training and fine-tuning, machine learning models can achieve high accuracy in sentiment analysis. By training on labelled data (reviews with known sentiment), models can learn to recognize and classify different sentiment expressions, such as positive, negative, or neutral.
3. **Flexibility:** Machine learning models can adapt and learn from new data, making them flexible for analyzing a variety of Amazon products and reviews. They can handle different languages, writing styles, and domains, making them versatile for sentiment analysis tasks.
4. **Feature extraction:** Machine learning algorithms can automatically extract relevant features from the text, such as keywords, phrases, or linguistic patterns, that contribute to sentiment. This eliminates the need for manual feature engineering, saving time and effort.

DISADVANTAGES:

1. **Data quality and bias:** Machine learning models heavily rely on the quality and representativeness of the training data. If the training data contains biases or is unrepresentative of the target audience, the model's predictions may be skewed or inaccurate.
2. **Contextual understanding:** Sentiment analysis is challenging because it requires understanding the context and nuances of language. Machine learning models may struggle to grasp

sarcasm, irony, or other forms of subtle sentiment expression, leading to misclassifications.

3. **Domain specificity:** Machine learning models trained on one domain may not perform as well on another. For example, a model trained on electronics reviews may not generalize well to book reviews. Domain-specific models need to be trained, which requires additional labelled data and effort.
4. **Interpretability:** Some machine learning models, such as deep learning models, can be considered black boxes, making it difficult to understand how and why a certain sentiment classification was made. This lack of interpretability can be problematic when trying to explain or justify the model's predictions.
5. **Data labeling and annotation:** Training machine learning models for sentiment analysis requires labeled data, which involves manual effort to annotate the sentiment of each review. This process can be time-consuming and expensive, especially when dealing with large datasets.

FUTURE SCOPE

Sentiment analysis is a valuable application of machine learning that can be used to analyze and understand customer opinions and sentiments expressed in reviews. Amazon, being a major e-commerce platform, can benefit from sentiment analysis to gain insights into customer satisfaction, identify potential issues, and make data-driven decisions to improve their products and services.

The future scope of Amazon review sentiment analysis using machine learning is promising. Here are a few potential areas of development and improvement:

1. **Enhanced accuracy**: Machine learning models for sentiment analysis can be further refined to improve accuracy. This can be achieved by incorporating more sophisticated algorithms, leveraging advanced natural language processing (NLP) techniques, and using larger and more diverse training datasets.
2. **Multilingual support**: Amazon operates in multiple countries and serves customers from various linguistic backgrounds. Expanding sentiment analysis to support multiple languages will enable Amazon to gain insights from customer reviews in different languages and cater to a more diverse customer base.
3. **Aspect-based sentiment analysis**: Traditional sentiment analysis typically focuses on overall sentiment polarity (positive, negative, or neutral). However, aspect-based sentiment analysis goes a step further by identifying sentiment towards specific aspects or features of a product. Implementing aspect-based sentiment analysis can provide more granular insights for product improvement and customer satisfaction.
4. **Real-time analysis**: Real-time sentiment analysis can provide immediate feedback on customer sentiments as reviews are posted. This enables Amazon to quickly identify and address any issues or concerns raised by customers, leading to improved customer service and satisfaction.
5. **Integration with customer support**: Integrating sentiment analysis with customer support systems can automate the analysis of customer feedback, allowing Amazon to prioritize and respond to critical issues more efficiently. This can enhance customer support processes and foster better customer relationships.
6. **Sentiment analysis of other data sources**: In addition to customer reviews, Amazon can explore sentiment analysis on other data sources,

such as social media posts, forums, and customer surveys. This broader analysis can provide a comprehensive understanding of customer sentiment across different channels.

7. **Recommender systems**: By incorporating sentiment analysis into their recommender systems, Amazon can provide more personalized recommendations based on customer preferences and sentiment towards specific products or categories.


Overall, the future scope of Amazon review sentiment analysis using machine learning involves refining the accuracy, expanding language support, implementing aspect-based analysis, enabling real-time analysis, integrating with customer support, analyzing other data sources, and incorporating sentiment analysis into recommender systems. These advancements will further enhance Amazon's ability to understand customer sentiments, improve customer satisfaction, and drive business growth.

ACTUAL CODE







FOR

SENTIMENT ANALYSIS OF AMAZON

REVIEWS

jupyter Sentiment Last Checkpoint: Last Thursday at 10:50 PM (autosaved)  Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel) O

      Code

```
In [211]: """Import required libraries"""

import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import nltk
nltk.download('stopwords')
from sklearn.preprocessing import StandardScaler
from sklearn.feature_extraction.text import CountVectorizer
from nltk.corpus import stopwords
from wordcloud import WordCloud

[nltk_data] Downloading package stopwords to
[nltk_data] C:\Users\HP\AppData\Roaming\nltk_data...
[nltk_data] Package stopwords is already up-to-date!
```

```
In [212]: #---Import the data set---
df=pd.read_csv("amazon.csv")
df.head()
```

Out[212]:

| Unnamed: 0 | reviewerName | overall | reviewText | reviewTime | day_diff | helpful_yes | helpful_no | total_vote | score_pos_neg_diff | score_average_rating | wilson_low |
|------------|--------------|----------|------------|---|------------|-------------|------------|------------|--------------------|----------------------|------------|
| 0 | 0 | NaN | 4 | No issues. | 23-07-2014 | 138 | 0 | 0 | 0 | 0 | 0.0 |
| 1 | 1 | 0mie | 5 | Purchased this for my device, it worked as adv... | 25-10-2013 | 409 | 0 | 0 | 0 | 0 | 0.0 |
| 2 | 2 | 1K3 | 4 | it works as expected, I should have sprung for... | 23-12-2012 | 715 | 0 | 0 | 0 | 0 | 0.0 |
| 3 | 3 | 1m2 | 5 | This think has worked out great Had a diff. br... | 21-11-2013 | 382 | 0 | 0 | 0 | 0 | 0.0 |
| 4 | 4 | 2&1/2Men | 5 | Bought it with Retail Packaging, arrived legit... | 13-07-2013 | 513 | 0 | 0 | 0 | 0 | 0.0 |

```
In [213]: """Preprocessing and Cleaning the reviews"""
#--Search for null vales--|
print(df.info())
print(df.shape)
print(df.isna().sum())

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4915 entries, 0 to 4914
Data columns (total 12 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Unnamed: 0             4915 non-null   int64
1   reviewerName           4914 non-null   object
2   overall                4915 non-null   int64
3   reviewText             4914 non-null   object
4   reviewTime             4915 non-null   object
5   day_diff               4915 non-null   int64
6   helpful_yes            4915 non-null   int64
7   helpful_no             4915 non-null   int64
8   total_vote             4915 non-null   int64
9   score_pos_neg_diff     4915 non-null   int64
10  score_average_rating   4915 non-null   float64
11  wilson_lower_bound     4915 non-null   float64
dtypes: float64(2), int64(7), object(3)
memory usage: 460.9+ KB
None
(4915, 12)
Unnamed: 0           0
reviewerName         1
overall              0
reviewText           1
reviewTime           0
day_diff             0
helpful_yes          0
helpful_no           0
total_vote           0
score_pos_neg_diff   0
score_average_rating 0
wilson_lower_bound   0
dtype: object
```

```
In [214]: df=df.dropna(axis=0)
print(df.shape)
print(df.isna().sum())
```

```
(4913, 12)
Unnamed: 0           0
reviewerName         0
overall              0
reviewText           0
reviewTime           0
day_diff             0
helpful_yes          0
helpful_no           0
total_vote           0
score_pos_neg_diff   0
score_average_rating 0
wilson_lower_bound   0
dtype: object
```

```
In [215]: df["overall"].value_counts().sort_index(ascending=False).plot(kind="bar",title='Count of Reviews by Stars')
plt.xlabel("Review Stars")
plt.show()
```



```
In [216]: #---Sentiment positive '+' or negative '-'---
df.loc[df['overall']<=3,'overall']=0
df.loc[df['overall']>3,'overall']=1
```

```
In [217]: #---Clean the review text column by removing the stopwords---
```

```
stp_words=stopwords.words('english')
def clean_review(review):
    cleanreview=" ".join(word for word in review.split() if word not in stp_words)
    return cleanreview
df['reviewText']=df['reviewText'].apply(clean_review)
#print(df['reviewText'].to_string(index=False))
df.reviewText.head(10)
```

```
Out[217]: 1 Purchased device, worked advertised. You never...
2 works expected. I sprung higher capacity. I th...
3 This think worked great.Had diff. bran 64gb ca...
4 Bought Retail Packaging, arrived legit, orange...
5 It's mini storage. It anything else supposed t...
6 I phone never skips beat. File transfers speed...
7 It's hard believe affordable digital become. 3...
8 Works HTC Rezound. Was running short space 64G...
9 galaxy s4, super fast card, totally happy, hap...
10 I like SD Card take music video downloads, per...
Name: reviewText, dtype: object
```

```
In [218]: """ Analysis of the data set """
```

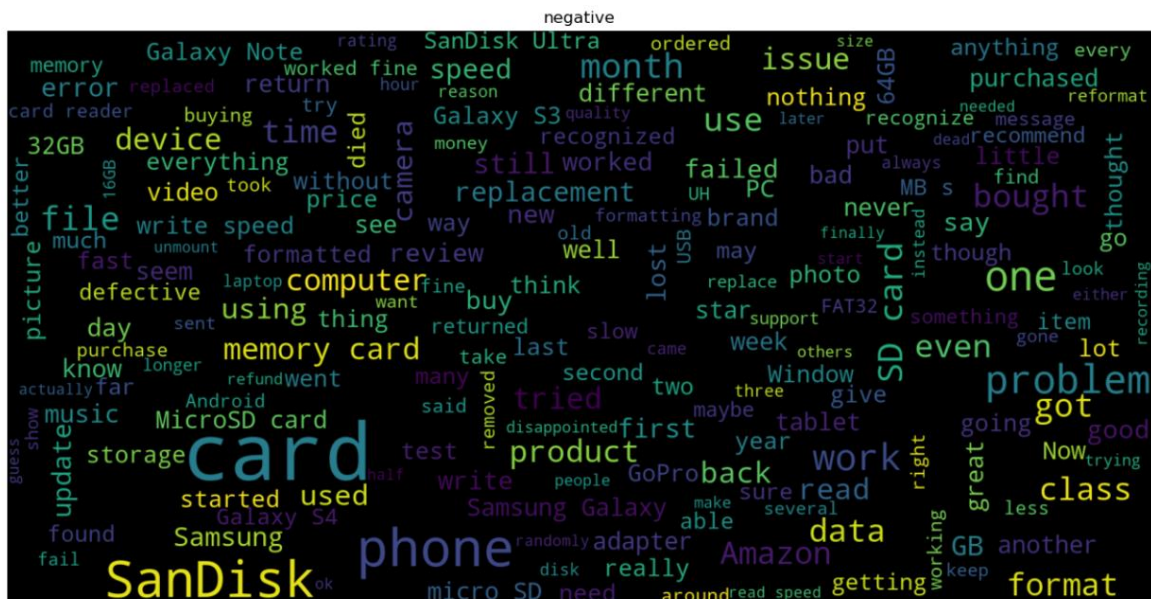
```
#---count the number of positive and negative sentiments---
print(df['overall'].value_counts())
```

| | |
|---|------|
| 1 | 4447 |
| 0 | 466 |

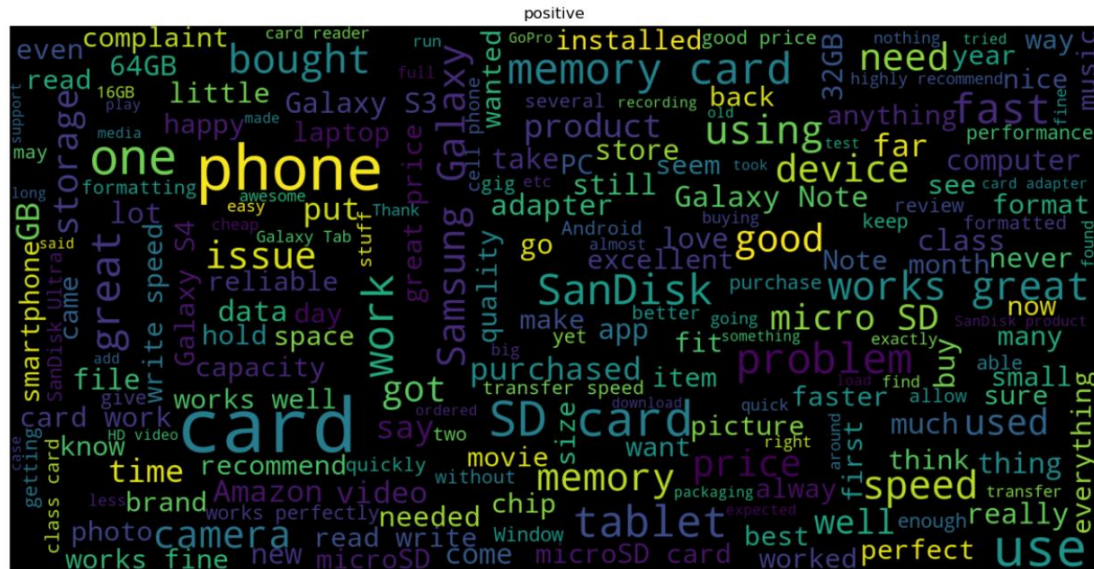
```
Name: overall, dtype: int64
```

```
In [219]: #---create the wordcloud with overall 0 or negative---
```

```
consolidated = ' '.join(word for word in df['reviewText'][df['overall']==0].astype(str))
wordCloud=WordCloud(width=1600,height=800,random_state=21,max_font_size=110)
plt.figure(figsize=(15,10))
plt.imshow(wordCloud.generate(consolidated),interpolation='bilinear')
plt.axis('off')
plt.title('negative')
plt.show()
```




```
In [220]: ##--create the wordcloud with overall 1 or positive ,---
consolidated=' '.join(word for word in df['reviewText'][(df['overall']==1).astype(str)])
wordcloud=wordcloud(width=1600,height=800,random_state=21,max_font_size=110)
plt.figure(figsize=(15,10))
plt.imshow(wordcloud.generate(consolidated),interpolation='bilinear')
plt.axis('off')
plt.title('positive')
plt.show()
```



```
In [221]: """Convert text into vector"""  
          #CountVectorizer  
  
          cv =CountVectorizer(max_features=2500)  
          x =cv.fit_transform(df['reviewText']).toarray()
```

```
In [222]: """Model training,evaluation and prediction"""

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,df['overall'],test_size=0.25 ,random state=0)
```

```
In [223]: ###-Logistic Regression---  
  
from sklearn.linear_model import LogisticRegression  
lr=LogisticRegression()  
  
#Model fitting  
lr.fit(x_train,y_train)  
  
#testing the model  
pred=lr.predict(x_test)  
# print(pred)
```



```
In [224]: #model accuracy
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
print("##### Logistic Regression USING VECTOR #####")
#--- Confusion Matrix---

print("Confusion Matrix")
matrix = confusion_matrix(y_test, pred)
print(matrix)

# Classification Report
print("\nClassification Report")
report = classification_report(y_test, pred)
print(report)

#---accuracy---
ac=accuracy_score(y_test,pred)
print('test accuracy: {:.2f}%'.format(ac*100))

##### Logistic Regression USING VECTOR #####
Confusion Matrix
[[ 68  40]
 [ 14 1107]]

Classification Report
              precision    recall  f1-score   support

     0       0.83        0.63       0.72         108
     1       0.97        0.99       0.98        1121

 accuracy          0.96         0.96         0.96        1229
 macro avg          0.90         0.81         0.85        1229
 weighted avg       0.95         0.96         0.95        1229

test accuracy: 95.61%
```

```
In [225]: from sklearn.naive_bayes import GaussianNB
gb=GaussianNB()
gb.fit(x_train,y_train)
y_pred=gb.predict(x_test)
```

```
In [226]: print("##### NAIVE BAYES USING VECTOR #####")
#--- Confusion Matrix---

print("Confusion Matrix")
matrix = confusion_matrix(y_test,y_pred)
print(matrix)

# Classification Report
print("\nClassification Report")
report = classification_report(y_test,y_pred)
print(report)

#---accuracy---
ac1=accuracy_score(y_test,y_pred)
print('test accuracy: {:.2f}%'.format(ac1*100))

##### NAIVE BAYES USING VECTOR #####
Confusion Matrix
[[ 47  61]
 [460 661]]

Classification Report
              precision    recall  f1-score   support

     0       0.09        0.44       0.15         108
     1       0.92        0.59       0.72        1121

 accuracy          0.58         0.58         0.58        1229
 macro avg          0.50         0.51         0.44        1229
 weighted avg       0.84         0.58         0.67        1229

test accuracy: 57.61%
```

```
In [227]: from sklearn.neighbors import KNeighborsClassifier
kn=KNeighborsClassifier()
kn.fit(x_train,y_train)
y_pred=kn.predict(x_test)
```

```
In [228]: print("##### KNN USING VECTOR #####")

#--- Confusion Matrix---

print("Confusion Matrix")
matrix = confusion_matrix(y_test,y_pred)
print(matrix)

# Classification Report
print("\nClassification Report")
report = classification_report(y_test,y_pred)
print(report)

#---accuracy---
ac2=accuracy_score(y_test,y_pred)
print('test accuracy: {:.2f}%'.format(ac2*100))
```

```
##### KNN USING VECTOR #####
Confusion Matrix
[[  4 104]
 [  2 119]]

Classification Report
precision    recall  f1-score   support

     0       0.67      0.04      0.07       108
     1       0.91      1.00      0.95      1121

 accuracy          0.91       1229
 macro avg          0.79      0.52      0.51       1229
weighted avg          0.89      0.91      0.88       1229

test accuracy: 91.38%
```

```
In [229]: from sklearn.tree import DecisionTreeClassifier
dc= DecisionTreeClassifier(criterion='gini',random_state=0)
dc.fit(x_train,y_train)
y_pred=dc.predict(x_test)
```

```
In [230]: print("##### DecisionTree USING VECTOR #####")

#--- Confusion Matrix---

print("Confusion Matrix")
matrix = confusion_matrix(y_test,y_pred)
print(matrix)

# Classification Report
print("\nClassification Report")
report = classification_report(y_test,y_pred)
print(report)

#---accuracy---
ac3=accuracy_score(y_test,y_pred)
print('test accuracy: {:.2f}%'.format(ac3*100))
```

```
##### DecisionTree USING VECTOR #####
Confusion Matrix
[[ 58  50]
 [ 69 1052]]

Classification Report
precision    recall  f1-score   support

     0       0.46      0.54      0.49       108
     1       0.95      0.94      0.95      1121

 accuracy          0.90       1229
 macro avg          0.71      0.74      0.72       1229
weighted avg          0.91      0.90      0.91       1229

test accuracy: 90.32%
```

```
In [231]: from sklearn.ensemble import RandomForestClassifier
rc= RandomForestClassifier(n_estimators=20,criterion='entropy')
rc.fit(x_train,y_train)
y_pred=rc.predict(x_test)
```

```
In [232]: print("##### RandomForest USING VECTOR #####")

#--- Confusion Matrix---

print("Confusion Matrix")
matrix = confusion_matrix(y_test,y_pred)
print(matrix)

# Classification Report
print("\nClassification Report")
report = classification_report(y_test,y_pred)
print(report)

#---accuracy---
ac4=accuracy_score(y_test,y_pred)
print('test accuracy: {:.2f}%'.format(ac4*100))

##### RandomForest USING VECTOR #####
Confusion Matrix
[[ 30  78]
 [ 9 112]]

Classification Report
      precision    recall  f1-score   support

     0       0.77     0.28     0.41       108
     1       0.93     0.99     0.96      1121

 accuracy          0.93       1229
 macro avg         0.85     0.63     0.69       1229
 weighted avg      0.92     0.93     0.91       1229

test accuracy: 92.92%
```

```
In [233]: """Convert text into scaler"""

#---Standard Scaler---

sc=StandardScaler()
x_train1=sc.fit_transform(x_train)
x_test1=sc.transform(x_test)
```

```
In [234]: kn=KNeighborsClassifier()

kn.fit(x_train1,y_train)
y_pred=kn.predict(x_test1)
```

```
In [235]: print("##### KNN USING STANDARED SCALER#####")

#--- Confusion Matrix---

print("Confusion Matrix")
matrix = confusion_matrix(y_test,y_pred)
print(matrix)

# Classification Report
print("\nClassification Report")
report = classification_report(y_test,y_pred)
print(report)

#---accuracy---
ac5=accuracy_score(y_test,y_pred)
print('test accuracy: {:.2f}%'.format(ac5*100))

##### KNN USING STANDARED SCALER#####
Confusion Matrix
[[ 1 107]
 [ 1 1120]]

Classification Report
      precision    recall  f1-score   support

     0       0.50     0.01     0.02       108
     1       0.91     1.00     0.95      1121

 accuracy          0.91       1229
 macro avg         0.71     0.50     0.49       1229
 weighted avg      0.88     0.91     0.87       1229

test accuracy: 91.21%
```

```
In [236]: gb=GaussianNB()
gb.fit(x_train1,y_train)
y_pred=gb.predict(x_test1)
```

```
In [237]: print("#####Naive Bayes USING STANDARD SCALER #####")
#--- Confusion Matrix---
print("Confusion Matrix")
matrix = confusion_matrix(y_test,y_pred)
print(matrix)

# Classification Report
print("\nClassification Report")
report = classification_report(y_test,y_pred)
print(report)

#---accuracy---
ac6=accuracy_score(y_test,y_pred)
print('test accuracy: {:.2f}%'.format(ac6*100))
```

#####Naive Bayes USING STANDARD SCALER #####

Confusion Matrix

```
[[ 47  61]
 [462 659]]
```

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 0.09 | 0.44 | 0.15 | 108 |
| 1 | 0.92 | 0.59 | 0.72 | 1121 |
| accuracy | | | 0.57 | 1229 |
| macro avg | 0.50 | 0.51 | 0.43 | 1229 |
| weighted avg | 0.84 | 0.57 | 0.67 | 1229 |

test accuracy: 57.45%

```
In [238]: dc= DecisionTreeClassifier(criterion='gini',random_state=0)
dc.fit(x_train1,y_train)
y_pred=dc.predict(x_test1)
```

```
In [239]: print("##### DecisionTree USING SCALER#####")
#--- Confusion Matrix---
print("Confusion Matrix")
matrix = confusion_matrix(y_test,y_pred)
print(matrix)

# Classification Report
print("\nClassification Report")
report = classification_report(y_test,y_pred)
print(report)

#---accuracy---
ac7=accuracy_score(y_test,y_pred)
print('test accuracy: {:.2f}%'.format(ac7*100))
```

DecisionTree USING SCALER#####

Confusion Matrix

```
[[ 59  49]
 [ 68 1053]]
```

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 0.46 | 0.55 | 0.50 | 108 |
| 1 | 0.96 | 0.94 | 0.95 | 1121 |
| accuracy | | | 0.90 | 1229 |
| macro avg | 0.71 | 0.74 | 0.72 | 1229 |
| weighted avg | 0.91 | 0.90 | 0.91 | 1229 |

test accuracy: 90.48%

```
In [240]: rc = RandomForestClassifier(n_estimators=20,criterion='entropy')
rc.fit(x_train1,y_train)
y_pred=rc.predict(x_test1)
```

```
In [242]: print("##### RandomForest USING SCALER #####")
#--- Confusion Matrix---
print("Confusion Matrix")
matrix = confusion_matrix(y_test,y_pred)
print(matrix)

# Classification Report
print("\nClassification Report")
report = classification_report(y_test,y_pred)
print(report)

#---accuracy---
ac8=accuracy_score(y_test,y_pred)
print('test accuracy: {:.2f}%'.format(ac8*100))
```

```
##### RandomForest USING SCALER #####
```

```
Confusion Matrix
[[ 27  81]
 [   7 1114]]
```

| Classification Report | precision | recall | f1-score | support |
|-----------------------|-----------|--------|----------|---------|
| 0 | 0.79 | 0.25 | 0.38 | 108 |
| 1 | 0.93 | 0.99 | 0.96 | 1121 |
| accuracy | | | 0.93 | 1229 |
| macro avg | 0.86 | 0.62 | 0.67 | 1229 |
| weighted avg | 0.92 | 0.93 | 0.91 | 1229 |

```
test accuracy: 92.84%
```

CONCLUSION

In conclusion, sentiment analysis for Amazon reviews using machine learning techniques can provide valuable insights into customer opinions and satisfaction levels. By analyzing the text of reviews, machine learning algorithms can classify them as positive, negative, or neutral, allowing businesses to understand customer sentiment towards their products or services.

Machine learning models trained on large datasets can accurately predict sentiment by learning patterns and relationships between words, phrases, and sentiment labels. These models can take into account various factors such as the overall rating, the language used, and specific keywords or phrases that indicate positive or negative sentiment.

By leveraging sentiment analysis, businesses can gain several benefits. They can identify common pain points or issues raised by customers and take necessary actions to address them. Additionally, they can track the impact of product changes or marketing campaigns on customer sentiment over time, allowing them to make data-driven decisions.

However, it's important to note that sentiment analysis models are not perfect and may have limitations. They might struggle with sarcasm, irony, or nuanced language, leading to misinterpretations. Additionally, the accuracy of sentiment analysis can vary depending on the quality and diversity of the training data and the complexity of the review content.

To enhance the accuracy of sentiment analysis, continuous training and fine-tuning of the machine learning models are necessary. Regularly updating the training data with new reviews and feedback can help improve the model's performance and ensure it remains relevant to changing customer sentiments.

Overall, sentiment analysis using machine learning for Amazon reviews is a powerful tool that can assist businesses in understanding and responding to customer feedback effectively. By leveraging this technology, businesses can improve customer satisfaction, make informed decisions, and enhance their overall product or service offerings.

From our analysis we observe that using different AI algorithms we get different accuracies. Using vector form of the data we used Logistic Regression, Naïve Bayes, KNN, Decision Tree, Random Forest models and out of all the models Logistic Regression gives the highest accuracy 95.61%.

And Using the scaler form of the data we used KNN, Naïve Bayes, Decision Tree, Random Forest and out of all the models Random Forest gives the highest accuracy 92.84%.

If we combine the results of both the vector and scaler Logistic Regression using vector gives the highest accuracy 95.61%.

BIBLIOGRAPHY

Source Code:

By using **Jupyter Notebook (Python 3.11.1)**

I Used:

- **Jupyter Notebook (Python 3.11.1) Software**
- **Internet Explorer**
- **Site:**

<https://www.geeksforgeeks.org/amazon-product-reviews-sentiment-analysis-in-python/>

<https://docs.aws.amazon.com/machine-learning/latest/dg/training-ml-models.html>

https://www.researchgate.net/publication/344869545_Sentiment_Analysis_on_Amazon_reviews

<https://www.diva-portal.org/smash/get/diva2:1241547/FULLTEXT01.pdf>

<https://www.repustate.com/blog/amazon-review-analysis/>

<https://www.kaggle.com/datasets/bittlingmayer/amazonreviews>

__THANK YOU__