Group Project Peer Evaluation Form

On a basis of 1 to 10, rate each group member's participation and contribution. List your detailed contribution

Yourself:

Name	Participation	Detailed Contribution
Sai Durgesh Mukkamala	10	graph analysis, introduction to the topic, literature review, Experimental Reviews

Other group members:

Name	Participation	Contribution
Uday kiran Gadde	10	Bio gram model , hyper parameter, conclusion
Gaurav Sai Mallaram	10	Project report, literature review, Power point

ABSTRACT

Sentiment analysis is a very popular technique for social network analysis. Sentiment analysis also termed as opinion mining is a process of automatically extracting knowledge from sentiments or opinions of others about some topic or problem. We can identify opinions in a large unstructured/structured data and analyze the polarity of opinions. Twitter is a large and rapidly growing micro blogging social networking website where people express their opinions in a short and simple manner of expressions. It is a common practice that merchants selling products on the Web ask their customers to review the products. In twitter number of customer reviews on different products is appearing. Mobile phones are a common domain in which number of customer reviews appears. This makes it difficult for a potential customer to read them in order to make a decision on whether to buy the product. We are only interested in the specific features of the phones that customers have opinions on and also whether the opinions are positive or negative. This project presents a lexicon-based approach for analyzing the customer reviews on mobile phone i.e., iPhone 14 over Twitter data to measure the popularity based on which the customer can decide whether to buy the product.

Keywords - Sentiment, opinion, score, Twitter, lexicon

Table of Contents

INTRODUCTION

CHAPTER-1

1.1 ABOUT THE PROJECT

There are various social media apps like twitter, Facebook, Instagram etc. and e commerce sites like flipkart, amazon which contain huge number of opinions and reviews about different products which people use in day today life. These sentiments will be very useful to the companies when the sentiments give some meaningful information about the status of their product or brand in the market. Companies will use those reviews to upgrade their product to a better version. Social media users use those sentiments to know about different opinions about products and do purchase accordingly.

Machine learning algorithms and lexicon-based approach can be used for sentiment analysis. Lexicon based approaches are too much dependent on the dictionaries. Machine learning approach has been proved to be better than the dictionary based. In machine learning, supervised algorithms are showing better classification results comparing to unsupervised or semi supervised algorithms.

Logistic Regression and Support vector machine is a supervised machine learning algorithm which proves to be a better one for doing sentiment analysis than the other algorithms. Live tweets have been scraped, pre-processed and then classified using logistic regression and SVM either as positive, negative or neural. Different kernel functions of SVM also have been used to know which kernel works better for classifying tweets.

1.2. PURPOSE

Most of the companies today improvise their products using different means. Reaching out to customers is a common means adopted by companies to understand the customer sentiments. Customer feedbacks are usually obtained using feedback forms, customer satisfaction surveys, reviews and activity tracking. Based on the data obtained from these feedbacks, the companies can improvise their

products and services associated with it. For example: e-commerce websites track the user click patterns and the recommendation engines suggest products accordingly. There are many tools available for sentiment analysis using the large data available on different social networking sites like Facebook, Twitter, LinkedIn etc.

Mostly sentiment analysis applications are used in prediction engines. For example: Election results prediction. These predictions are usually done by building a model on the data. Sentiment analysis has been previously implemented in customer feedback system, prediction engines and recommendation engines. This paper presents a novel lexicon-based sentiment analysis technique which is used to analyze the sentiment of people on different mobile phones. The user selects the input on the mobile phones and the brand of the device via radio buttons, and drop-down menus respectively. Based on the input, graph is created showing the sentiment score distribution of each mobile phone brand.

1.3. MOTIVATION

Sentiment analysis is a technique of analyzing the opinions commented in social media on various topics. There are few ways the sentiment analysis can be done. Machine learning plays a crucial role for analyzing the opinions and reviews. Mobile related tweets have been scraped from twitter. Noise on the tweets has been removed using pre-processing and feature vectors were created. Support Vector Machine has been used to classify the reviews as either positive or negative. 4 cross validation technique were used to bring out the better accuracy. 3 different sizes of dataset with iphone11 reviews have been used for training and testing with different kernels of SVM. RBF kernel is found to be working.

1.4. SCOPE

This application will predict the sentiments for the iPhone 14 tweets which are posted by the customer on twitter. Sentiments can be classified into positive, negative or neutral. Firstly, we have performed data preprocessing steps in order to

refine the data. Secondly, we have generated the sentiments for the tweets. Then we have built the model using logistic regression and support vector machine. We have also computed the respective accuracies and confusion matrix.

SYSTEM STUDY AND ANALYSIS

CHAPTER-2

2.1. FEASIBILITY STUDY

Preliminary investigation examines project feasibility, the likelihood the application will be useful to the user. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging traditional desktop centric applications and porting them to mobile devices. All systems are feasible if they are given unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

- Operation Feasibility
- Technical Feasibility
- Economic Feasibility

2.1.1 Operation Feasibility

The operational issue usually raised during the feasibility stage of the investigation includes the following:

- **User-friendly**

  Client will use the python resources for screens of their various transactions i.e., we have jupyter notebook for loading the iPhone 14 tweets dataset, performed data preprocessing techniques, generated sentiments based on the tweets which are classified as positive, negative or neutral.

- **Security**

  Python Production Server uses HTTPS for security. You can configure the security of a server instance to be as broad or specific as required. The instance can simply encrypt the communication channel between it and a client or it can block unauthorized clients from accessing applications.

- **Availability**

This software will be available since it is maintained at one place.

2.1.2. Technical Feasibility

The technical issue raised during the feasibility stage of the investigation includes the following:

- Does the necessary technology exist to do what is suggested?
- Do the proposed equipment's have the technical capacity to hold the data required to use the new system?
- Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
- Can the system be upgraded if developed?

As part of achieving technical competency for developing the proposed system users are required to acquire skill set in python.

2.1.3. Economic Feasibility

Usage of Loopback by connecting frontend and backend for communication ensures less cost. Open-source technologies like python usage minimizes the cost for the Developer.

2.2EXISTING SYSTEM

The corpus used [8] was gathered from Twitter, tweets about top 10 automotive brands. Feature vector is the most important concept in implementing a classifier. Unigram model outperforms all other models. The choice of kernel and proper tuning of SVM hyper parameters are core factors, contributing to SVM accuracy. RBF kernel is used and the SVM type used is C-SVC which is the multiclass classification. The dataset consisted [9] Epinions.com movie reviews half positive and half negative. SVM was used to bring several favourability measures (OS good, Turney) for phrases and adjectives which is combined with unigram

models and lemmatized versions of unigram models. . Linear kernel SVM Used 3 and 10 cross fold validation. Lemmas outperform unigrams in all experiments.

2.3. PROPOSED SYSTEM

In proposed system, an attention-based long short-term memory model to predict stock price trend. The model consists of four parts: data loading from kaggle, data preprocessing, generating sentiments, model training, computing accuracy and confusion matrix.

This application will predict the sentiments for the iPhone 14 tweets which are posted by the customer on twitter. Sentiments can be classified into positive, negative or neutral. Firstly, we have performed data preprocessing steps in order to refine the data. Secondly, we have generated the sentiments for the tweets. Then we have built the model using logistic regression and vector machine. We have also computed the respective accuracies and confusion matrix.

SYSTEM REQUIREMENTS

CHAPTER-3

3.1 REQUIREMENTS SPECIFICATION

A software requirement specification (SRS) is a detailed description of a software system to be developed with its functional and non-functional requirements. The SRS is developed based the agreement between the customers and contractors. It may include the use cases of how user is going to interact with software system. The software requirement specification document consistent of all necessary requirements required for project development. To develop the software system, we should have clear understanding of software system. To achieve this, we need to continuous communication with customers to gather all requirements.

A good SRS defines the how software system will interact with all internal modules, hardware, communication with other programs and human user interactions with wide range of real-life scenarios. Using the software requirements specification (SRS) document on QA lead, managers create test plan.

3.1.1. Types

There are two types of requirements specification. They are:

- Functional requirements specification
- Non-functional requirements specification

3.1.1.1. Functional Requirements Specification

A Functional Requirement (FR) is a description of the service that the software must offer. It describes a software system or its component. A function is nothing but inputs to the software system, its behavior, and outputs. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform. Functional Requirements are also called Functional Specification.

In software engineering and systems engineering, a Functional Requirement can range from the high-level abstract statement of the sender's necessity to detailed mathematical functional requirement specifications. Functional software requirements help you to capture the intended behavior of the system.

The present application has been divided in to following modules.

- Dataset Loading
- Data Preprocessing
- Model Building
- Plotting graphs and computing accuracy

3.1.1.2. Non- Functional Requirement Specifications

Non-functional requirement (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of non-functional requirement, "how fast does the website load?" Failing to meet non-functional requirements can result in systems that fail to satisfy user needs.

Non-functional requirement allows you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users is > 10000. Description of non-functional requirements is just as critical as a functional requirement.

Important to User	Important to Team	Important to business
Performance	Maintainability	Time to market
Security	Portability	Cost
Usability	Reusability	Flexibility
Compatibility	Testability	Speed
Accessibility	Naming Convention	
Flexibility	Tech Stack	
Disaster Recovery	Monitoring	

3.1.1.2. Non-functional requirements

- **Performance**

The system must be interactive and the delays involved must be less. So, in every action-response of the system, there are no immediate delays. In case of opening windows forms, of popping error messages and saving the settings or sessions there is delay much below 2 seconds. In case of opening databases, sorting questions and evaluation there are no delays and the operation is performed in less than 2 seconds for opening, sorting, computing, posting > 95% of the files. Also, when connecting to the server the delay is based editing on the distance of the 2 systems and the Configuration between them so there is high probability that there will be or not a successful connection in less than 20 seconds for sake of good communication.

- **Reliability**

As the system provide the right tools for discussion, problem solving it must be made sure that the system is reliable in its operations and for securing the sensitive details.

- **Safety**

Information transmission should be securely transmitted to server without any changes in information.

- **Security**

The main security concern is for users account hence proper login mechanism should be used to avoid hacking. The tablet id registration is way

to spam check for increasing the security. Hence, security is provided from unwanted use of recognition software.

- **Availability**

If the internet service gets disrupted while sending information to the server, the information can be sent again for verification.

- **Usability**

As the system is easy to handle and navigates in the most expected way with no delays. In that case the system program reacts accordingly and transverses quickly between its states.

- **Portability**

It is the usability of the same software in different environments. The pre -requirement for portability is the generalized abstraction between the application logic and system interfaces. When software with the same functionality is produced for several computing platforms, portability is the key issue for development cost reduction.

- **Testability**

Software testability is the degree to which a software artifact (i.e., a software system, software module, requirements- or design document) supports testing in a given test context. If the testability of the software artifact is high, then finding faults in the system (if it has any) by means of testing is easier.

3.2. SOFTWARE REQUIREMENTS

Operating System : Windows7 & above versions

Coding Language : Python and Anaconda and jupyter

SYSTEM DESIGN

CHAPTER-4

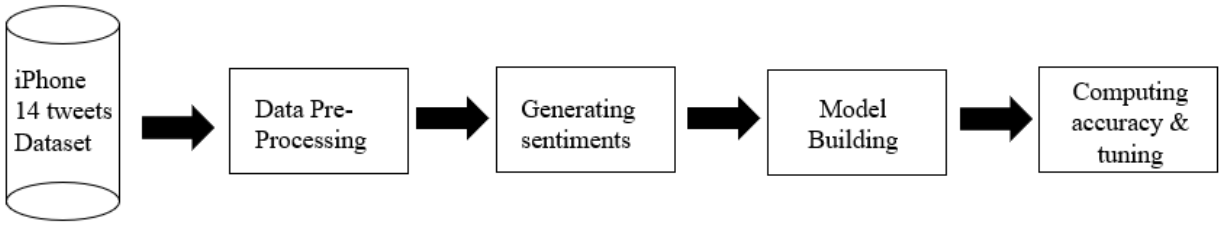
4.1 INTRODUCTION

System design is the process or art of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development. There is some overlap and synergy with the disciplines of systems analysis, systems architecture and systems engineering.

4.2 SYSTEM ARCHITECTURE

In proposed system, an model is built to generate sentiments for iPhone 14 tweets by the customers. The model consists of four parts: data loading from kaggle, data preprocessing, generating sentiments, model training, computing accuracy and confusion matrix.

This application will predict the sentiments for the iPhone 14 tweets which are posted by the customer on twitter. Sentiments can be classified into positive, negative or neutral. Firstly, we have performed data preprocessing steps in order to refine the data. Secondly, we have generated the sentiments for the tweets. Then we have built the model using logistic regression and vector machine. We have also computed the respective accuracies and confusion matrix.



SOFTWARE ENVIRONMENT

CHAPTER-5

5.1 INTRODUCTION

Software environment emerged in the middle of the midrange era as a means of improving software quality and productivity through automation. A software environment may be described as an ‘operating system environment and a collection of tools or subroutines. A slightly better definition of software environment is a ‘coordinated collection of software tools organized to support some approach to software development or conform to some software process model’, where software tools are defined as ‘computer programs that assist engineers with the design and development of computer-based systems.

Structured programming environments were created as a means of improving software reliability and productivity using guidelines, code libraries, structured coding, top-down development, chief programmer teams, standards, procedures, documentation, education and metrics. Software factories were soon created to introduce discipline and repeatability, software visualization tools, the capture of customer needs or requirements, automated software testing and software reuse. Computer-assisted software engineering or CASE was also created to enhance software productivity and reliability by automating document production, diagram design, code compilation, software testing, configuration management, management reporting and sharing of data by multiple developers.

5.2. ABOUT PYTHON

Python has become one of the most popular programming languages in the world in recent years. It's used in everything from machine learning to building websites and software testing. It can be used by developers and non-developers alike.

Python, one of the most popular programming languages in the world, has created everything from Netflix’s recommendation algorithm to the software that controls self-driving cars. Python is a general-purpose language, which means it’s

designed to be used in a range of applications, including data science, software and web development, automation, and generally getting stuff done.

5.2.1. What is Python?

Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems. This versatility, along with its beginner-friendliness, has made it one of the most-used programming languages today. A survey conducted by industry analyst firm RedMonk found that it was the second-most popular programming language among developers in 2021.

5.2.2. What is Python used for?

Python is commonly used for developing websites and software, task automation, data analysis, and data visualization. Since it's relatively easy to learn, Python has been adopted by many non-programmers such as accountants and scientists, for a variety of everyday tasks, like organizing finances.

"Writing programs is a very creative and rewarding activity," says University of Michigan and Coursera instructor Charles R Severance in his book *Python for Everybody*. "You can write programs for many reasons, ranging from making your living to solving a difficult data analysis problem to having fun to helping someone else solve a problem."

What can you do with python?

Some things include:

- Data analysis and machine learning
- Web development
- Automation or scripting
- Software testing and prototyping
- Everyday tasks

5.2.3. Data analysis and machine learning

Python has become a staple in data science, allowing data analysts and other professionals to use the language to conduct complex statistical calculations, create data visualizations, build machine learning algorithms, manipulate and analyze data, and complete other data-related tasks.

Python can build a wide range of different data visualizations, like line and bar graphs, pie charts, histograms, and 3D plots. Python also has a number of libraries that enable coders to write programs for data analysis and machine learning more quickly and efficiently, like TensorFlow and Keras.

6.2.4. Web development

Python is often used to develop the back end of a website or application—the parts that a user doesn't see. Python's role in web development can include sending data to and from servers, processing data and communicating with databases, URL routing, and ensuring security. Python offers several frameworks for web development. Commonly used ones include Django and Flask.

Some web development jobs that use Python include back-end engineers, full stack engineers, Python developers, software engineers, and DevOps engineers.

5.2.5. Automation or scripting

If you find yourself performing a task over and over again, you could work more efficiently by automating it with Python. Writing code used to build these automated processes is called scripting. In the coding world, automation can be used to check for errors across multiple files, convert files, execute simple math, and remove duplicates in data.

Python can even be used by relative beginners to automate simple tasks on the computer—such as renaming files, finding and downloading online content or sending emails or texts at desired intervals.

5.2.6. Software testing and prototyping

In software development, Python can aid in tasks like build control, bug tracking, and testing. With Python, software developers can automate testing for new products or features. Some Python tools used for software testing include Green and Requestium.

SYSTEM IMPLEMENTATION

CHAPTER-6

6.1. INTRODUCTION

Systems implementation is the process of -

1. defining how the information system should be built (i.e., physical system design),
2. ensuring that the information system is operational and used,
3. ensuring that the information system meets quality standard (i.e., quality assurance).

6.2. MODULES AND DESCRIPTION

In proposed system, a model is built to generate sentiments for iPhone 14 tweets by the customers. The model consists of four parts: data loading from kaggle, data preprocessing, generating sentiments, model training, computing accuracy and confusion matrix.

This application will predict the sentiments for the iPhone 14 tweets which are posted by the customer on twitter. Sentiments can be classified into positive, negative or neutral. Firstly, we have performed data preprocessing steps in order to refine the data. Secondly, we have generated the sentiments for the tweets. Then we have built the model using logistic regression and vector machine. We have also computed the respective accuracies and confusion matrix.

6.3. SAMPLE CODE

6.3.1 Introduction

Coding is the process of designing, writing, testing, debugging, and maintaining the source code of computer programs. This source code is written in one or more programming languages. The purpose of programming is to create a set of instructions that computers use to perform specific operations or to exhibit desired behaviors. The process of writing source code often requires expertise in many different subjects, including knowledge of the application domain, specialized algorithms and formal logic.

7.3.2 Coding

```
### Data Collection
import pandas as pd
import numpy as np
import re
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import style
style.use('ggplot')
from textblob import TextBlob
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
from wordcloud import WordCloud
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix,
ConfusionMatrixDisplay

pd.read_csv('iphone14-query-tweets.csv')
df.head()
df.info()
df.isnull().sum()
df.columns
text_df = df.drop(['date_time', 'username', 'user_location', 'user_description',
      'verified', 'followers_count', 'following_count', 'tweet_like_count',
      'tweet_retweet_count', 'tweet_reply_count', 'source'], axis=1)
text_df.head()
print(text_df['tweet_text'].iloc[0],"\n")
print(text_df['tweet_text'].iloc[1],"\n")
print(text_df['tweet_text'].iloc[2],"\n")
print(text_df['tweet_text'].iloc[3],"\n")
print(text_df['tweet_text'].iloc[4],"\n")

def data_processing(text):
    text = text.lower()
    text = re.sub(r"https\S+|www\S+https\S+", ",text, flags=re.MULTILINE)
    text = re.sub(r'\@w+|\#',",text)
    text = re.sub(r'[^\w\s]',",text)
    text_tokens = word_tokenize(text)
    filtered_text = [w for w in text_tokens if not w in stop_words]
    return " ".join(filtered_text)

text_df.tweet_text = text_df['tweet_text'].apply(data_processing)
text_df = text_df.drop_duplicates('tweet_text')
```

```
stemmer = PorterStemmer()
def stemming(data):
    text = [stemmer.stem(word) for word in data]
    return data
text_df['tweet_text'] = text_df['tweet_text'].apply(lambda x: stemming(x))
text_df.head()

print(text_df['tweet_text'].iloc[0],"\n")
print(text_df['tweet_text'].iloc[1],"\n")
print(text_df['tweet_text'].iloc[2],"\n")
print(text_df['tweet_text'].iloc[3],"\n")
print(text_df['tweet_text'].iloc[4],"\n")

text_df.info()
def polarity(text):
    return TextBlob(text).sentiment.polarity
text_df['polarity'] = text_df['tweet_text'].apply(polarity)
text_df.head(10)

def sentiment(label):
    if label <0:
        return "Negative"
    elif label ==0:
        return "Neutral"
    elif label>0:
        return "Positive"
text_df['sentiment'] = text_df['polarity'].apply(sentiment)
text_df.head()

fig = plt.figure(figsize=(5,5))
sns.countplot(x='sentiment', data = text_df)
fig = plt.figure(figsize=(7,7))
colors = ("yellowgreen", "gold", "red")
wp = {'linewidth':2, 'edgecolor':"black"}
tags = text_df['sentiment'].value_counts()
explode = (0.1,0.1,0.1)
tags.plot(kind='pie', autopct='%1.1f%%', shadow=True, colors = colors,
          startangle=90, wedgeprops = wp, explode = explode, label='')
plt.title('Distribution of sentiments')
pos_tweets = text_df[text_df.sentiment == 'Positive']
pos_tweets = pos_tweets.sort_values(['polarity'], ascending= False)
pos_tweets.head()

text = ' '.join([word for word in pos_tweets['tweet_text']])
plt.figure(figsize=(20,15), facecolor='None')
wordcloud = WordCloud(max_words=500, width=1600, height=800).generate(text)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.title('Most frequent words in positive tweets', fontsize=19)
plt.show()
neg_tweets = text_df[text_df.sentiment == 'Negative']
```

```
neg_tweets = neg_tweets.sort_values(['polarity'], ascending= False)
neg_tweets.head()

text = ' '.join([word for word in neg_tweets['tweet_text']])
plt.figure(figsize=(20,15), facecolor='None')
wordcloud = WordCloud(max_words=500, width=1600, height=800).generate(text)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.title('Most frequent words in negative tweets', fontsize=19)
plt.show()
neutral_tweets = text_df[text_df.sentiment == 'Neutral']
neutral_tweets = neutral_tweets.sort_values(['polarity'], ascending= False)
neutral_tweets.head()

text = ' '.join([word for word in neutral_tweets['tweet_text']])
plt.figure(figsize=(20,15), facecolor='None')
wordcloud = WordCloud(max_words=500, width=1600, height=800).generate(text)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.title('Most frequent words in neutral tweets', fontsize=19)
plt.show()
[{"metadata":{},"cell_type":"markdown","source":"### Create a Bigram Model"}]

feature_names = vect.get_feature_names()
print("Number of features: {}\n".format(len(feature_names)))
print("First 20 features:\n {}".format(feature_names[:20]))
X = text_df['tweet_text']
Y = text_df['sentiment']
X = vect.transform(X)
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
print("Size of x_train:", (x_train.shape))
print("Size of y_train:", (y_train.shape))
print("Size of x_test:", (x_test.shape))
print("Size of y_test:", (y_test.shape))
import warnings
warnings.filterwarnings('ignore')
logreg = LogisticRegression()
logreg.fit(x_train, y_train)
logreg_pred = logreg.predict(x_test)
logreg_acc = accuracy_score(logreg_pred, y_test)
print("Test accuracy: {:.2f}%".format(logreg_acc*100))
print(confusion_matrix(y_test, logreg_pred))
print("\n")
print(classification_report(y_test, logreg_pred))
style.use('classic')
cm = confusion_matrix(y_test, logreg_pred, labels=logreg.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix = cm, display_labels=logreg.classes_)
disp.plot()


from sklearn.model_selection import GridSearchCV
```

```
param_grid={'C':[0.001, 0.01, 0.1, 1, 10]}
grid = GridSearchCV(LogisticRegression(), param_grid)
grid.fit(x_train, y_train)
print("Best parameters:", grid.best_params_)
y_pred = grid.predict(x_test)
logreg_acc = accuracy_score(y_pred, y_test)
print("Test accuracy: {:.2f}%".format(logreg_acc*100))
print(confusion_matrix(y_test, y_pred))
print("\n")
print(classification_report(y_test, y_pred))

from sklearn.svm import LinearSVC
SVCmodel = LinearSVC()
SVCmodel.fit(x_train, y_train) svc_pred = SVCmodel.predict(x_test)
svc_acc = accuracy_score(svc_pred, y_test)
print("test accuracy: {:.2f}%".format(svc_acc*100))
print(confusion_matrix(y_test, svc_pred))
print("\n")
print(classification_report(y_test, svc_pred))
grid = {
    'C':[0.01, 0.1, 1, 10],
    'kernel':["linear","poly","rbf","sigmoid"],
    'degree':[1,3,5,7],
    'gamma':[0.01,1]
}
grid = GridSearchCV(SVCmodel, param_grid)
grid.fit(x_train, y_train)
print("Best parameter:", grid.best_params_)
y_pred = grid.predict(x_test)
logreg_acc = accuracy_score(y_pred, y_test)
print("Test accuracy: {:.2f}%".format(logreg_acc*100))
print(confusion_matrix(y_test, y_pred))
print("\n")
print(classification_report(y_test, y_pred))
```

TESTING

CHAPTER-7

7.1. INTRODUCTION

Testing is a process of executing a program with the aim of finding error. To make our software perform well it should be error free. If testing is done successfully, it will remove all the errors from the software.

7.2. TESTING

Testing can also be stated as the process of verifying and validating that a software or application is bug free, meets the technical requirements as guided by its design and development, and meets the user requirements effectively and efficiently with handling all the exceptional and boundary cases.

Principles of Testing:

(i) All the test should meet the customer requirements.
(ii) To make our software testing should be performed by third party.
(iii) Exhaustive testing is not possible. As we need the optimal amount of testing based on the risk assessment of the application.
(iv) All the test to be conducted should be planned before implementing it.
(v) It follows pareto rule (80/20 rule) which states that 80% of errors comes from 20% of program components.
(vi) Start testing with small parts and extend it to large parts.

Testing Objectives:

The following are the testing objectives....

- Testing is a process of executing a program with the intent of finding an error.
- A good test case is one that has a high probability of finding an as yet undiscovered error.
- A successful test is one that uncovers as a yet undiscovered error.

Steps:

Testing can be divided into two steps. They are –

- **Verification**

  It refers to the set of tasks that ensure that software correctly implements a specific function.

- **Validation**

  It refers to a different set of tasks that ensure that the software that has been built is traceable to customer requirements.

Verification: "Are we building the product, right?"
Validation: "Are we building the right product?"

Testing Techniques:

1) White Box Testing

White box testing is a test case design method that uses the control structure of the procedural design to derive test cases. After performing white box testing it was identified that

- The E-health care system guarantees that all independent paths within the modules have been exercised at least once.
- It has been exercised all logical decisions on their true and false sides.

2)Black Box Testing:

Black box tests are designed to uncover errors in functional requirements without regard to the internal workings of a program. Black box testing techniques focus on the information domain of the software.

Black box testing attempts to find errors in the following categories.

- Incorrect or missing functions
- Interface errors
- Errors in the data structures or external database access
- Performance errors

Types of Testing:

Testing can be broadly classified into two types:

- **Manual Testing**

  Manual testing includes testing a software manually, i.e., without using any automated tool or any script. In this type, the tester takes over the role of an end-user and tests the software to identify any unexpected behavior or bug.

- **Automation Testing**

  Automation testing, which is also known as Test Automation, is when the tester writes scripts and uses another software to test the product.

Levels of Testing:

Testing levels can be majorly classified into 4 levels. They are -

- **Unit Testing**

  A level of the software testing process where individual units/components of a software/system are tested. The purpose is to validate that each unit of the software performs as designed.

- **Integration Testing**

  A level of the software testing process where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units.

- **System Testing**

  A level of the software testing process where a complete, integrated system/software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements.

- **Acceptance Testing**

  A level of the software testing process where a system is tested for acceptability. The purpose of this test is to evaluate the system's compliance with the business.

EXPERIMENTAL RESULTS

CHAPTER-8

8.1. EXPERIMENTAL RESULTS

Sentimental Analysis Using Customer Twitter Tweets on Iphone 14

Twitter sentiment analysis is performed to identify the sentiments of the people towards various topics. For this project, we will be analysing the sentiment of people towards Pfizer vaccines.

We will be using the data available on Kaggle to create this machine learning model. The collected tweets from Twitter will be analysed using machine learning to identify the different sentiments present in the tweets. The different sentiments identified in this project include positive sentiment, negative sentiment and neutral sentiment. We will also be using different classifiers to see which classifier gives the best model accuracy.

Importing Libraries

In [4]:
```
import pandas as pd
import numpy as np
import re
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import style
style.use('ggplot')
from textblob import TextBlob
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
from wordcloud import WordCloud
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, ConfusionMatrixDisplay
```

Reading the Dataset and Analyzing the dataset

In [5]:
```
df = pd.read_csv('iphone14-query-tweets.csv')
```

In [6]:
```
df.head()
```

Out[6]:

	date_time	username	user_location	user_description	verified	followers_count	following_count	tweet_like_count	tweet_retweet_count	twee
0	2022-09-08 22:49:29+00:00	TheAppleGang101	NaN	A new account dedicated to all of the latest I...	False	10	28	0	0	
1	2022-09-08 22:49:27+00:00	TheJessicats	1999	Tweet like nobody's reading • standup comedian...	False	1642	1444	0	0	
2	2022-09-08 22:49:16+00:00	itschefnotjeff	📌 mom's basement	🖼️🗂️🖥️🚀	False	77	87	0	0	
3	2022-09-08 22:49:09+00:00	HalfRonin	Between the darkness and light	Preferring to be the dumbest person in the roo...	False	549	717	0	0	
4	2022-09-08 22:49:09+00:00	Deejayrayman	Texas	Father of 3. Follower of Christ. Lover of all ...	False	48	153	0	0	

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144245 entries, 0 to 144244
Data columns (total 12 columns):
 #   Column               Non-Null Count   Dtype 
---  ------               --------------   ----- 
 0   date_time            144245 non-null  object
 1   username             144245 non-null  object
 2   user_location        97925 non-null   object
 3   user_description     127678 non-null  object
 4   verified             144245 non-null  bool  
 5   followers_count      144245 non-null  int64 
 6   following_count      144245 non-null  int64 
 7   tweet_like_count     144245 non-null  int64 
 8   tweet_retweet_count  144245 non-null  int64 
 9   tweet_reply_count    144245 non-null  int64 
 10  source               144245 non-null  object
 11  tweet_text           144245 non-null  object
dtypes: bool(1), int64(5), object(6)
memory usage: 12.2+ MB
```

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
date_time                 0
username                  0
user_location         46320
user_description      16567
verified                  0
followers_count           0
following_count           0
tweet_like_count          0
tweet_retweet_count       0
```

In [9]:
```
df.columns
```

Out[9]:
```
Index(['date_time', 'username', 'user_location', 'user_description',
       'verified', 'followers_count', 'following_count', 'tweet_like_count',
       'tweet_retweet_count', 'tweet_reply_count', 'source', 'tweet_text'],
      dtype='object')
```

In [11]:
```
text_df = df.drop(['date_time', 'username', 'user_location', 'user_description',
       'verified', 'followers_count', 'following_count', 'tweet_like_count',
       'tweet_retweet_count', 'tweet_reply_count', 'source'], axis=1)
text_df.head()
```

Out[11]:

	tweet_text
0	iPhone SE 3 gets more expensive in the UK afte...
1	@Travon I hope it happens before my iPhone 14 ...
2	Nah iphone 14 upgrade just not it, change my m...
3	To the shock of absolutely no one: \n\nApple: ...
4	@TMobile @TMobileHelp will you be able to orde...

In [12]:
```
print(text_df['tweet_text'].iloc[0],"\n")
print(text_df['tweet_text'].iloc[1],"\n")
print(text_df['tweet_text'].iloc[2],"\n")
print(text_df['tweet_text'].iloc[3],"\n")
print(text_df['tweet_text'].iloc[4],"\n")
```

```
iPhone SE 3 gets more expensive in the UK after the iPhone 14 event https://t.co/08sJ28TZ2W 

@Travon I hope it happens before my iPhone 14 arrives. 
```

Data Processing

In [17]:
```
def data_processing(text):
    text = text.lower()
    text = re.sub(r"https\S+|www\S+https\S+", '',text, flags=re.MULTILINE)
    text = re.sub(r'\@w+|\#','',text)
    text = re.sub(r'[^\w\s]','',text)
    text_tokens = word_tokenize(text)
    filtered_text = [w for w in text_tokens if not w in stop_words]
    return " ".join(filtered_text)
```

In [19]:
```
text_df.tweet_text = text_df['tweet_text'].apply(data_processing)
```

In [20]:
```
text_df = text_df.drop_duplicates('tweet_text') #Removing the duplicates from the dataset
```

In [21]:
```
stemmer = PorterStemmer()
def stemming(data):
    text = [stemmer.stem(word) for word in data]
    return data
```

In [23]:
```
text_df['tweet_text'] = text_df['tweet_text'].apply(lambda x: stemming(x))
```

In [24]:
```
text_df.head()
```

Out[24]:

	tweet_text
0	iphone se 3 gets expensive uk iphone 14 event
1	travon hope happens iphone 14 arrives
2	nah iphone 14 upgrade change mind
3	shock absolutely one apple ready iphone 14 3 p...
4	tmobile tmobilehelp able order iphone 14 pro o...

In [25]:
```
print(text_df['tweet_text'].iloc[0],"\n")
print(text_df['tweet_text'].iloc[1],"\n")
print(text_df['tweet_text'].iloc[2],"\n")
print(text_df['tweet_text'].iloc[3],"\n")
print(text_df['tweet_text'].iloc[4],"\n")
```

```
iphone se 3 gets expensive uk iphone 14 event 

travon hope happens iphone 14 arrives 

nah iphone 14 upgrade change mind 

shock absolutely one apple ready iphone 14 3 ppl left care yeah see also iphone 13 3 wut see also iphone 12 3 want go back e
ven 

tmobile tmobilehelp able order iphone 14 pro online tomorrow phone 
```

In [26]:
```
text_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 108538 entries, 0 to 144243
Data columns (total 1 columns):
 #   Column      Non-Null Count   Dtype 
---  ------      --------------   ----- 
 0   tweet_text  108538 non-null  object
dtypes: object(1)
memory usage: 1.7+ MB
```

In [27]:
```
def polarity(text):
    return TextBlob(text).sentiment.polarity
```

In [28]:
```
text_df['polarity'] = text_df['tweet_text'].apply(polarity)
```

In [29]:
```
text_df.head(10)
```

Out[29]:

	tweet_text	polarity
0	iphone se 3 gets expensive uk iphone 14 event	-0.500000
1	travon hope happens iphone 14 arrives	0.000000
2	nah iphone 14 upgrade change mind	0.000000
3	shock absolutely one apple ready iphone 14 3 p...	0.100000
4	tmobile tmobilehelp able order iphone 14 pro o...	0.500000
5	chinas four major carriers support us version ...	0.062500
6	apple set unveil iphone 14 gear	0.000000
7	iphone 14 proslightly used ashaiman washington...	0.000000

In [30]:
```
def sentiment(label):
    if label <0:
        return "Negative"
    elif label ==0:
        return "Neutral"
    elif label>0:
        return "Positive"
```

In [31]:
```
text_df['sentiment'] = text_df['polarity'].apply(sentiment)
```

In [32]:
```
text_df.head()
```

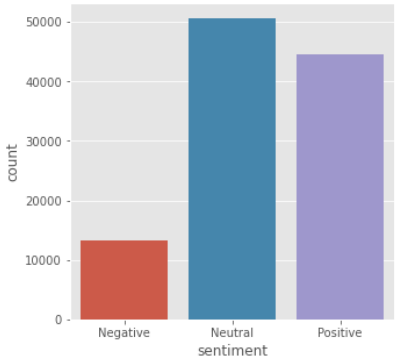
Out[32]:

	tweet_text	polarity	sentiment
0	iphone se 3 gets expensive uk iphone 14 event	-0.5	Negative
1	travon hope happens iphone 14 arrives	0.0	Neutral
2	nah iphone 14 upgrade change mind	0.0	Neutral
3	shock absolutely one apple ready iphone 14 3 p...	0.1	Positive
4	tmobile tmobilehelp able order iphone 14 pro o...	0.5	Positive

Plotting the graph for sentiments generated from the dataset

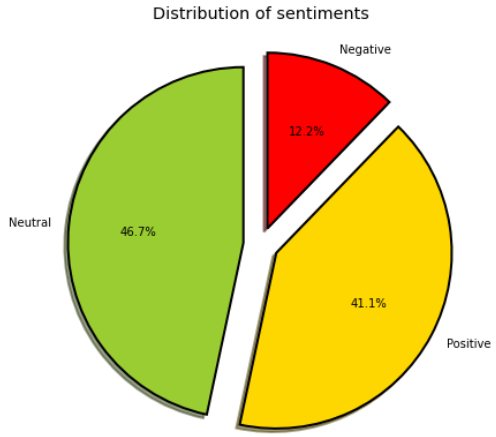
In [33]:
```
fig = plt.figure(figsize=(5,5))
sns.countplot(x='sentiment', data = text_df)
```

Out[33]: <AxesSubplot:xlabel='sentiment', ylabel='count'>



In [35]:
```
fig = plt.figure(figsize=(7,7))
colors = ("yellowgreen", "gold", "red")
wp = {'linewidth':2, 'edgecolor':"black"}
tags = text_df['sentiment'].value_counts()
explode = (0.1,0.1,0.1)
tags.plot(kind='pie', autopct='%1.1f%%', shadow=True, colors = colors,
          startangle=90, wedgeprops = wp, explode = explode, label='')
plt.title('Distribution of sentiments')
```

Out[35]: Text(0.5, 1.0, 'Distribution of sentiments')

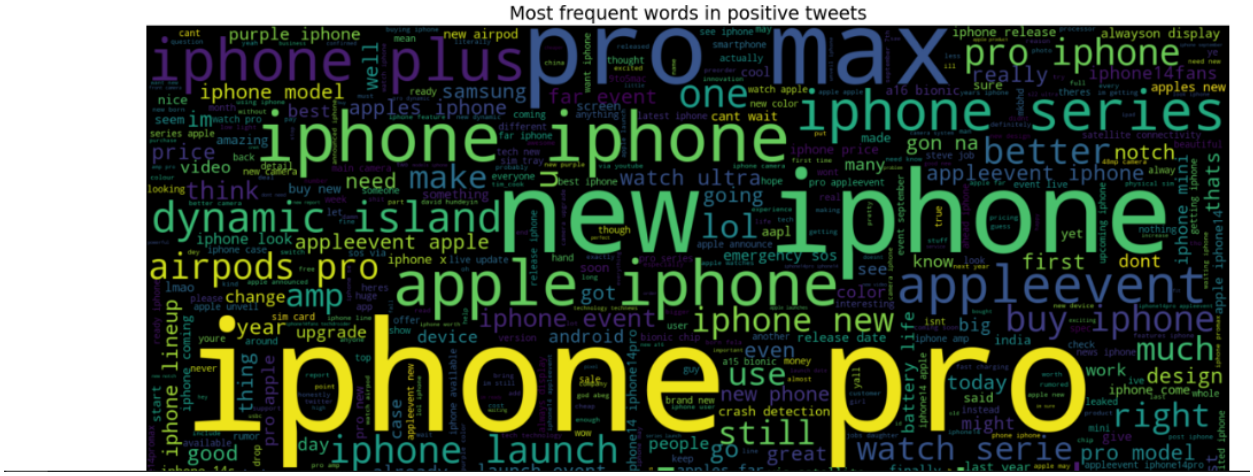


In [36]:
```
pos_tweets = text_df[text_df.sentiment == 'Positive']
pos_tweets = pos_tweets.sort_values(['polarity'], ascending= False)
pos_tweets.head()
```

Out[36]:

	tweet_text	polarity	sentiment
41451	iphone 14 pro max best phone ever made	1.0	Positive
108219	best iphone cases 13 pro iphone 14 cases starb...	1.0	Positive
93463	iphone14fans looks totally awesome	1.0	Positive
96348	ishankotwal7 lordschott iphone14fans samsung b...	1.0	Positive
99710	iphone14fans tried generate iphone 14 stable d...	1.0	Positive

In [38]:
```
text = ' '.join([word for word in pos_tweets['tweet_text']])
plt.figure(figsize=(20,15), facecolor='None')
wordcloud = WordCloud(max_words=500, width=1600, height=800).generate(text)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.title('Most frequent words in positive tweets', fontsize=19)
plt.show()
```



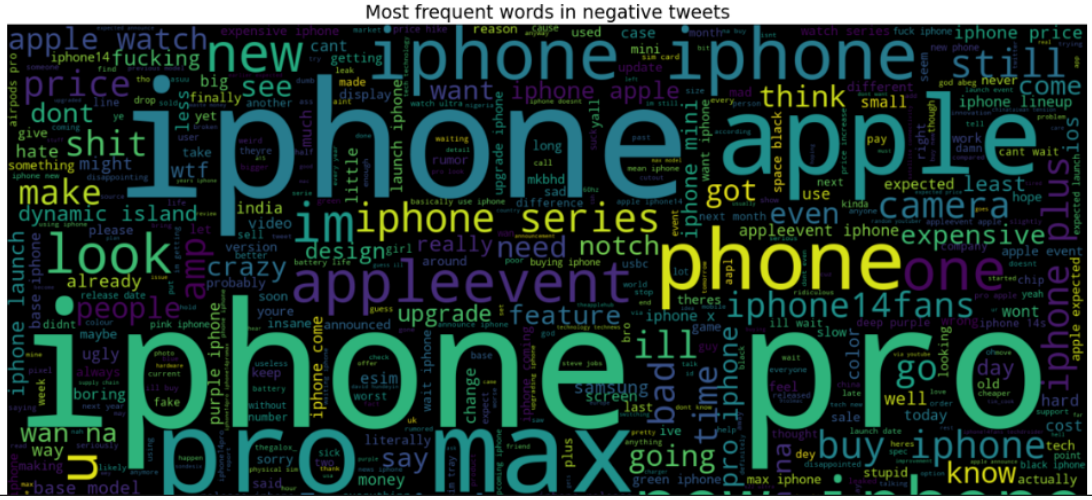
In [39]:
```
neg_tweets = text_df[text_df.sentiment == 'Negative']
neg_tweets = neg_tweets.sort_values(['polarity'], ascending= False)
neg_tweets.head()
```

Out[39]:

	tweet_text	polarity	sentiment
133893	far still consistently expecting launch ios 16...	-5.551115e-18	Negative
63710	nuxavi_ thegeeksjournai iphone 14 basic 128gb ...	-5.551115e-18	Negative
108236	radical_dude711 7feeling good schools coming a...	-9.251859e-18	Negative
10855	ahhh galaxys display completely broken ahhh al...	-1.110223e-17	Negative
16532	im also genuinely disappointed see space grey ...	-1.387779e-17	Negative

In [40]:
```
text = ' '.join([word for word in neg_tweets['tweet_text']])
plt.figure(figsize=(20,15), facecolor='None')
wordcloud = WordCloud(max_words=500, width=1600, height=800).generate(text)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.title('Most frequent words in negative tweets', fontsize=19)
plt.show()
```

Most frequent words in negative tweets



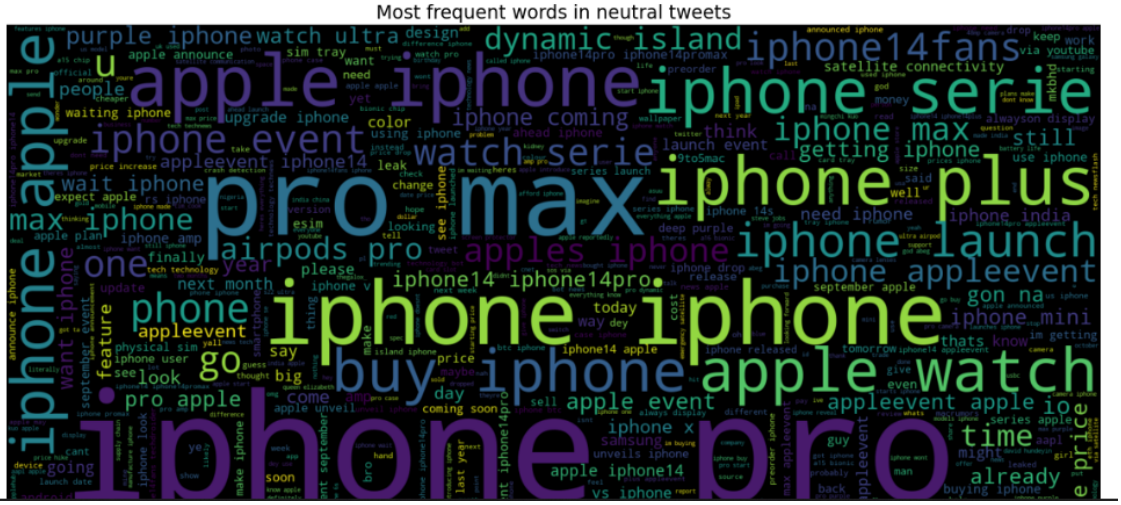
In [41]:
```
neutral_tweets = text_df[text_df.sentiment == 'Neutral']
neutral_tweets = neutral_tweets.sort_values(['polarity'], ascending= False)
neutral_tweets.head()
```

Out[41]:

	tweet_text	polarity	sentiment
1	travon hope happens iphone 14 arrives	0.0	Neutral
94581	iphone14fans nothing phone 1	0.0	Neutral
94584	iphone 14 models apple iphone14 iphone	0.0	Neutral
94586	iphone 14 satellite connectivity rumor came back	0.0	Neutral
94587	andobil car vent phone mount 2022 upgrade stur...	0.0	Neutral

In [42]:
```
text = ' '.join([word for word in neutral_tweets['tweet_text']])
plt.figure(figsize=(20,15), facecolor='None')
wordcloud = WordCloud(max_words=500, width=1600, height=800).generate(text)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.title('Most frequent words in neutral tweets', fontsize=19)
plt.show()
```

Most frequent words in neutral tweets



Create a Bigram Model ¶

In [43]:
```
vect = CountVectorizer(ngram_range=(1,2)).fit(text_df['tweet_text'])
```

In [44]:
```
feature_names = vect.get_feature_names()
print("Number of features: {}\n".format(len(feature_names)))
print("First 20 features:\n {}".format(feature_names[:20]))
```

```
C:\Users\HP\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function get_feature_names is deprec
ated; get_feature_names is deprecated in 1.0 and will be removed in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

```
Number of features: 499743

First 20 features:
 ['00', '00 bottom', '00 iphone', '000', '000 6k', '000 appleevent', '000 entrylevel', '000 good', '000 help', '000 retwee
t', '000 travel', '000 tree', '000 usd', '000 weekend', '0000', '0000 0200', '000000', '000000 units', '000000000010', '0000
01s']
```

In [45]:
```
X = text_df['tweet_text']
Y = text_df['sentiment']
X = vect.transform(X)
```

In [46]:
```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
```

In [47]:
```
print("Size of x_train:", (x_train.shape))
print("Size of y_train:", (y_train.shape))
print("Size of x_test:", (x_test.shape))
print("Size of y_test:", (y_test.shape))
```

```
Size of x_train: (86830, 499743)
Size of y_train: (86830,)
Size of x_test: (21708, 499743)
Size of y_test: (21708,)
```

In [48]:
```
import warnings
warnings.filterwarnings('ignore')
```

Build Logistic Regression Model

In [49]:
```
logreg = LogisticRegression()
logreg.fit(x_train, y_train)
logreg_pred = logreg.predict(x_test)
logreg_acc = accuracy_score(logreg_pred, y_test)
print("Test accuracy: {:.2f}%".format(logreg_acc*100))
```

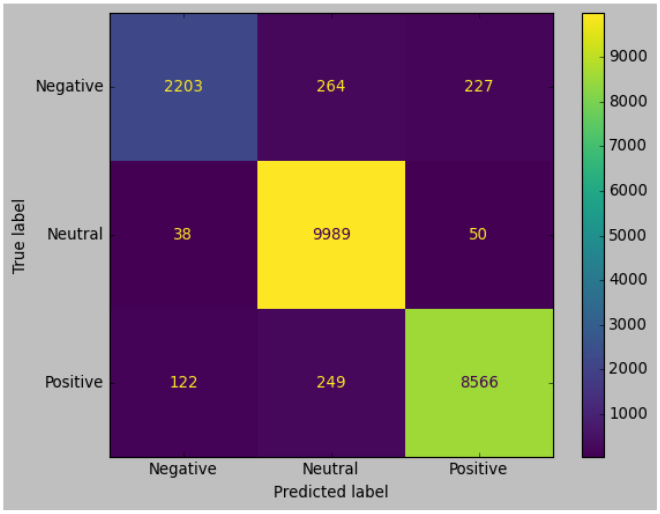
```
Test accuracy: 95.62%
```

In [50]:
```
print(confusion_matrix(y_test, logreg_pred))
print("\n")
print(classification_report(y_test, logreg_pred))
```

```
[[2203  264  227]
 [  38 9989   50]
 [ 122  249 8566]]


              precision    recall  f1-score   support

    Negative       0.93      0.82      0.87      2694
     Neutral       0.95      0.99      0.97     10077
    Positive       0.97      0.96      0.96      8937

    accuracy                           0.96     21708
   macro avg       0.95      0.92      0.94     21708
weighted avg       0.96      0.96      0.96     21708
```

In [51]:
```
style.use('classic')
cm = confusion_matrix(y_test, logreg_pred, labels=logreg.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix = cm, display_labels=logreg.classes_)
disp.plot()
```

Out[51]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1978948a670>



Perform Hyperparameter tuning Using GridSearchCV

Hyperparameter tuning consists of finding a set of optimal hyperparameter values for a learning algorithm while applying this optimized algorithm to any data set. That combination of hyperparameters maximizes the model's performance, minimizing a predefined loss function to produce better results with fewer errors.

In [52]:
```
from sklearn.model_selection import GridSearchCV
```

In [65]:
```
param_grid={'C':[0.001, 0.01, 0.1, 1, 10]}
grid = GridSearchCV(LogisticRegression(), param_grid)
grid.fit(x_train, y_train)
```

Out[65]:
```
GridSearchCV(estimator=LogisticRegression(),
             param_grid={'C': [0.001, 0.01, 0.1, 1, 10]})
```

In [54]:
```
print("Best parameters:", grid.best_params_)
```

```
Best parameters: {'C': 10}
```

In [55]:
```
y_pred = grid.predict(x_test)
```

In [56]:
```
logreg_acc = accuracy_score(y_pred, y_test)
print("Test accuracy: {:.2f}%".format(logreg_acc*100))
```

```
Test accuracy: 95.97%
```

In [57]:
```
print(confusion_matrix(y_test, y_pred))
print("\n")
print(classification_report(y_test, y_pred))
```

```
[[2266  220  208]
 [  42 9983   52]
 [ 134  218 8585]]


              precision    recall  f1-score   support

    Negative       0.93      0.84      0.88      2694
     Neutral       0.96      0.99      0.97     10077
    Positive       0.97      0.96      0.97      8937

    accuracy                           0.96     21708
   macro avg       0.95      0.93      0.94     21708
weighted avg       0.96      0.96      0.96     21708
```

Build Support Vector Machine (SVM) Model

In [58]:
```
from sklearn.svm import LinearSVC
```

In [59]:
```
SVCmodel = LinearSVC()
SVCmodel.fit(x_train, y_train)
```

Out[59]:
```
LinearSVC()
```

In [60]:
```
svc_pred = SVCmodel.predict(x_test)
svc_acc = accuracy_score(svc_pred, y_test)
print("test accuracy: {:.2f}%".format(svc_acc*100))
```

```
test accuracy: 96.84%
```

In [61]:
```
print(confusion_matrix(y_test, svc_pred))
print("\n")
print(classification_report(y_test, svc_pred))
```

```
[[ 2356   148   190]
 [   41 10001    35]
 [  129   144  8664]]


              precision    recall  f1-score   support

    Negative       0.93      0.87      0.90      2694
     Neutral       0.97      0.99      0.98     10077
    Positive       0.97      0.97      0.97      8937

    accuracy                           0.97     21708
   macro avg       0.96      0.95      0.95     21708
weighted avg       0.97      0.97      0.97     21708
```

In [62]:
```
grid = {
    'C':[0.01, 0.1, 1, 10],
    'kernel':["linear","poly","rbf","sigmoid"],
    'degree':[1,3,5,7],
    'gamma':[0.01,1]
}
grid = GridSearchCV(SVCmodel, param_grid)
grid.fit(x_train, y_train)
```

Out[62]:
```
GridSearchCV(estimator=LinearSVC(), param_grid={'C': [0.001, 0.01, 0.1, 1, 10]})
```

In [63]:
```
print("Best parameter:", grid.best_params_)
```

```
Best parameter: {'C': 10}
```

In [69]:
```
y_pred = grid.predict(x_test)
```

In [70]:
```
logreg_acc = accuracy_score(y_pred, y_test)
print("Test accuracy: {:.2f}%".format(logreg_acc*100))
```

```
Test accuracy: 95.97%
```

In [71]:

```
print(confusion_matrix(y_test, y_pred))
print("\n")
print(classification_report(y_test, y_pred))
```

```
[[2266  220  208]
 [  42 9983   52]
 [ 134  218 8585]]


              precision    recall  f1-score   support

    Negative       0.93      0.84      0.88      2694
     Neutral       0.96      0.99      0.97     10077
    Positive       0.97      0.96      0.97      8937

    accuracy                           0.96     21708
   macro avg       0.95      0.93      0.94     21708
weighted avg       0.96      0.96      0.96     21708
```

CONCLUSION

CHAPTER-9

Conclusion:

In proposed system, a model is built to generate sentiments for iPhone 14 tweets by the customers. The model consists of four parts: data loading from kaggle, data preprocessing, generating sentiments, model training, computing accuracy and confusion matrix.

This application will predict the sentiments for the iPhone 14 tweets which are posted by the customer on twitter. Sentiments can be classified into positive, negative or neutral. Firstly, we have performed data preprocessing steps in order to refine the data. Secondly, we have generated the sentiments for the tweets. Then we have built the model using logistic regression and vector machine. We have also computed the respective accuracies and confusion matrix.

Accuracy obtained for logistic regression is 95.62%. On tuning it with GridSearchCV, the accuracy obtained is 95.97%. Accuracy obtained for SVM is 96.84%. On tuning it with GridSearchCV, the accuracy obtained is 95.97%.

THANK YOU

.