Table of Contents

List of Figures

CHAPTER 1: MACHINE LEARNING

1.1. INTRODUCTION:

Machine Learning(ML) is the scientific study of algorithms and statistical models thatcomputer systems use in order to perform a specific task effectively without using explicitinstructions, relying on patterns and inference instead. It is seen as a subset of Artificial Intelligence(AI).

1.2 IMPORTANCE OF MACHINE LEARNING:

Consider some of the instances where machine learning is applied: the self-driving Googlecar, cyber fraud detection, online recommendation engines—like friend suggestions on Facebook, Netflix showcasing the movies and shows you might like, and “more items to consider” and “get yourself a little something” on Amazon—are all examples of applied machine learning. All these examples echo the vital role machine learning has begun to take intoday’s data-rich world[1].

Machines can aid in filtering useful pieces of information that help in major advancements,and we are already seeing how this technology is being implemented in a wide variety of industries.

With the constant evolution of the field, there has been a subsequent rise in the uses, demands, and importance of machine learning. Big data has become quite a buzzword in the lastfew years; that’s in part due to increased sophistication of machine learning, which helps analyse those big chunks of big data. Machine learning has also changed the way data extraction, and interpretation is done by involving automatic sets of generic methods that havereplaced traditional statistical techniques. The process flow depicted here represents how machine learning works.

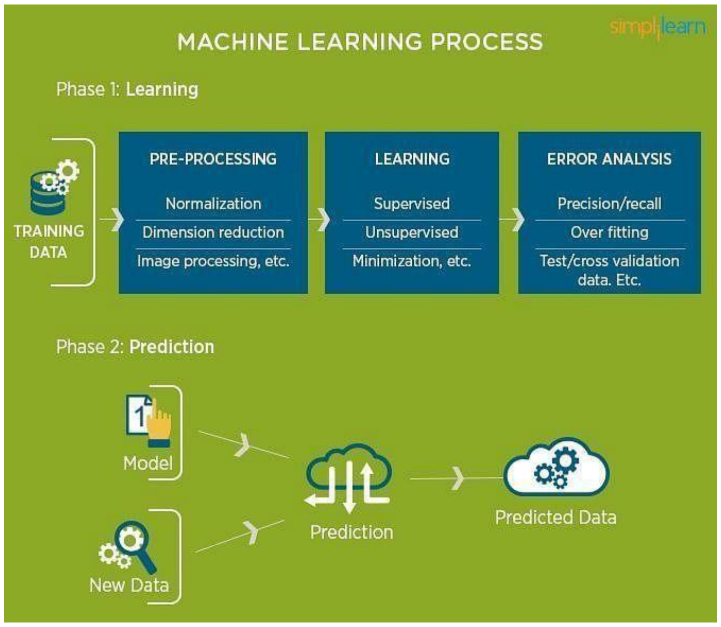


Fig 1.2.1: The Process Flow

1.3 USES OF MACHINE LEARNING:

Earlier in this article, we mentioned some applications of machine learning. To understand the concept of machine learning better, let’s consider some more examples: web search results,real-time ads on web pages and mobile devices, email spam filtering, network intrusion detection, and pattern and image recognition. All these are by-products of applying machine learning to analyze huge volumes of data.

Traditionally, data analysis was always characterized by trial and error, an approach that becomes impossible when data sets are large and heterogeneous[1]. Machine learning comes as thesolution to all this chaos by proposing clever alternatives to analyzing huge volumes of data.

By developing fast and efficient algorithms and data-driven models for real-time processingof data, machine learning can produce accurate results and analysis.

1.4 TYPES OF LEARNING ALGORITHMS:

The types of machine learning algorithms differ in their approach, the type of data they inputand output, and the type of task or problem that they are intended to solve.

1.4.1 Supervised Learning:

When an algorithm learns from example data and associated target responses that can consist of numeric values or string labels, such as classes or tags, in order to later predict thecorrect response when posed with new examples comes under the category of supervised learning.

Supervised machine learning algorithms uncover insights, patterns, and relationships from alabeled training dataset – that is, a dataset that already contains a known value for the target variable for each record. Because you provide the machine learning algorithm with the correct answers for a problem during training, it is able to “learn” how the rest of the features relate to the target, enabling you to uncover insights and make predictions about future outcomes based on historical data.

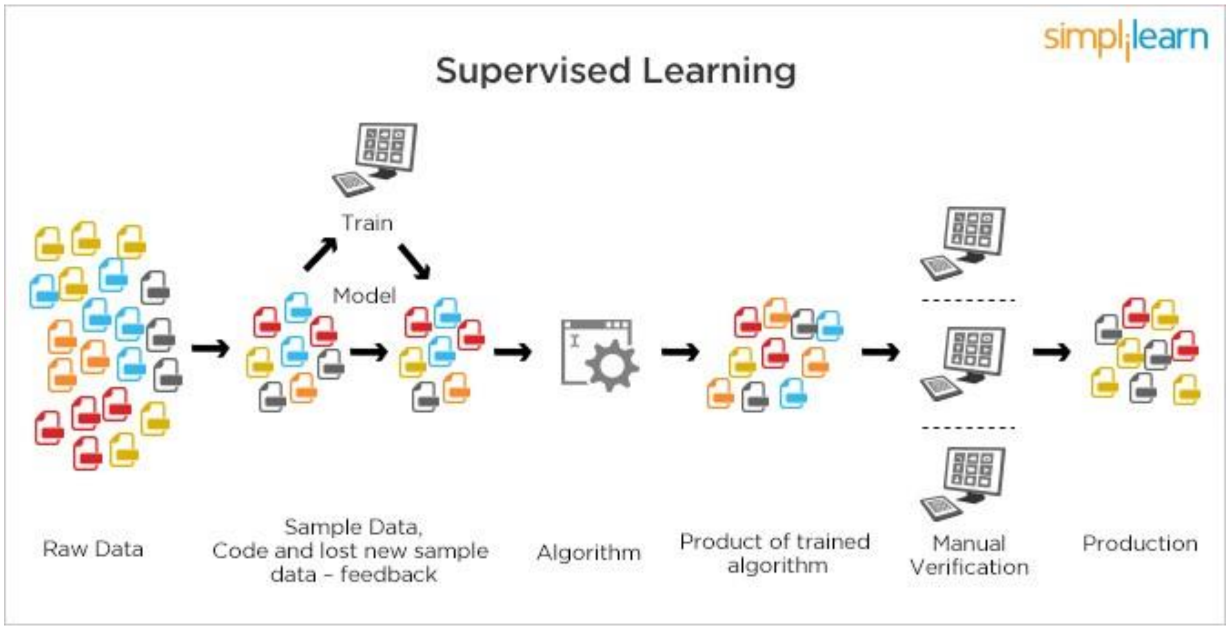


Fig 1.4.1: Supervised Learning

1.4.2 Unsupervised Learning:

When an algorithm learns from plain examples without any associated response, leaving to the algorithm to determine the data patterns on its own. This type of algorithm tends to restructure the data into something else, such as new features that may represent a class or a newseries of uncorrelated values[1]. They are quite useful in providing humans with insights into the meaning of data and new useful inputs to supervised machine learning algorithms.

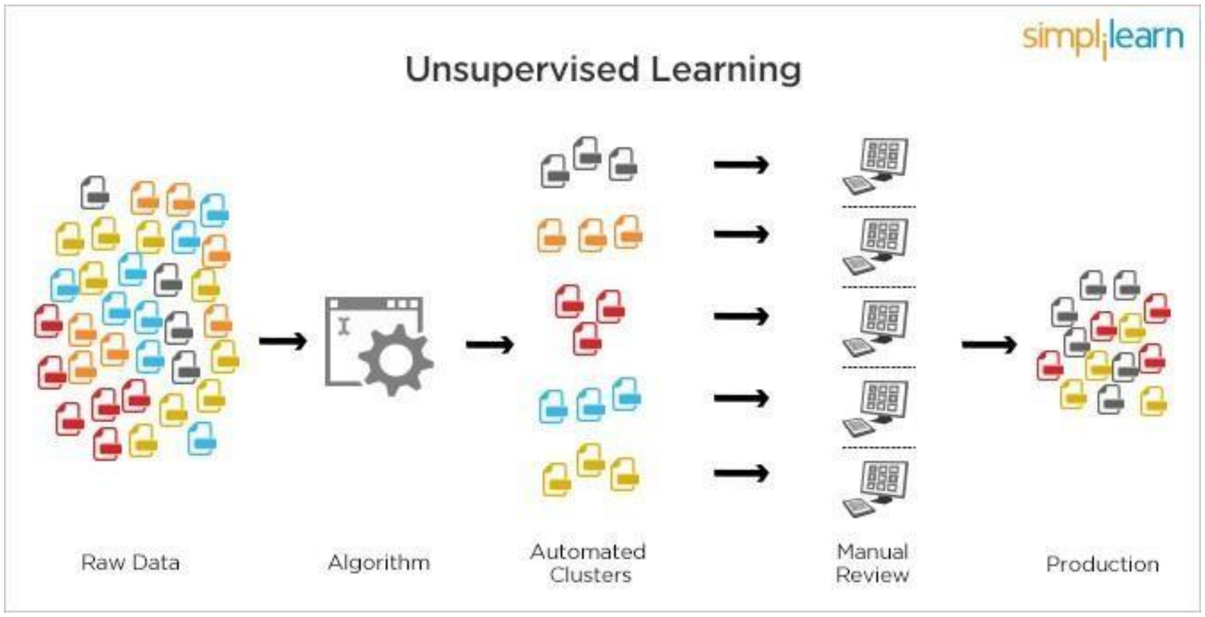


Fig 1.4.2: Unsupervised Learning

Popular techniques where unsupervised learning is used also include self-organizing maps, nearest neighbor mapping, singular value decomposition, and k-means clustering. Basically, online recommendations, identification of data outliers, and segment text topics are all examples of unsupervised learning.

1.4.3 Semi Supervised Learning:

As the name suggests, semi-supervised learning is a bit of both supervised and unsupervisedlearning and uses both labeled and unlabelled data for training. In a typical scenario, the algorithm would use a small amount of labeled data with a large amount of unlabelled data.

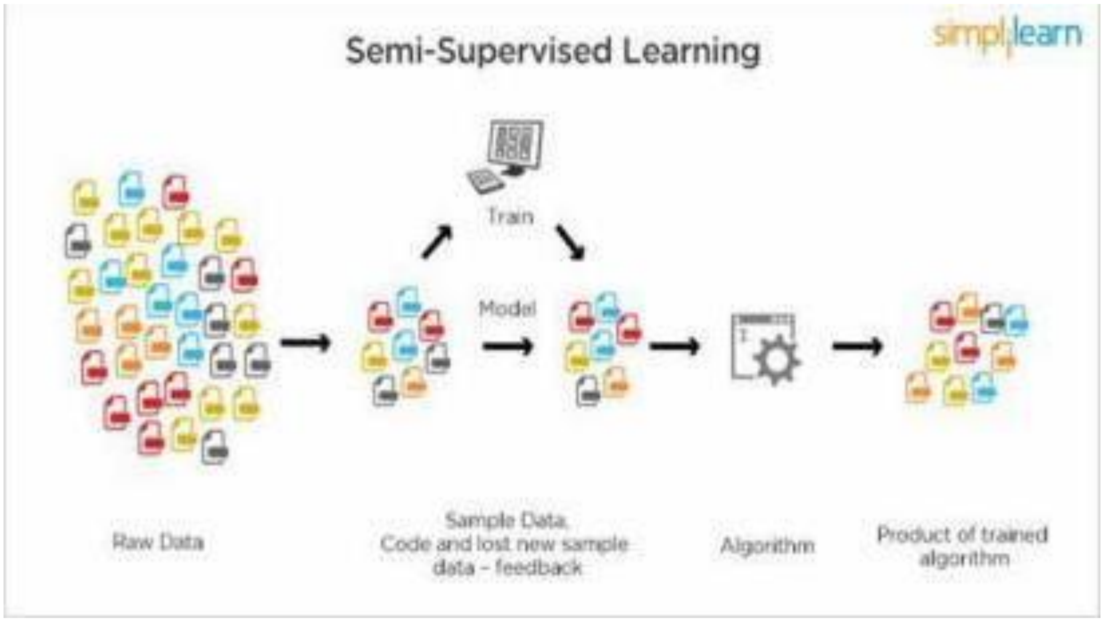


Fig 1.4.3: Semi Supervised Learning

CHAPTER 2: PYTHON

Basic programming language used for machine learning is: PYTHON

2.1. INTRODUCTION TO PYTHON:

· Python is a high-level, interpreted, interactive and object-oriented scripting language.
· Python is a general-purpose programming language that is often applied in scripting roles
· Python is Interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is like PERL and PHP. · Python is Interactive: You can 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 ofprogramming that encapsulates code within objects.

2.1.1 History of python:

Python was developed by GUIDO VAN ROSSUM in early 1990's · Its latest version is 3.7.3, itis generally called as python3[2].

2.2 HOW TO SETUP PYTHON:

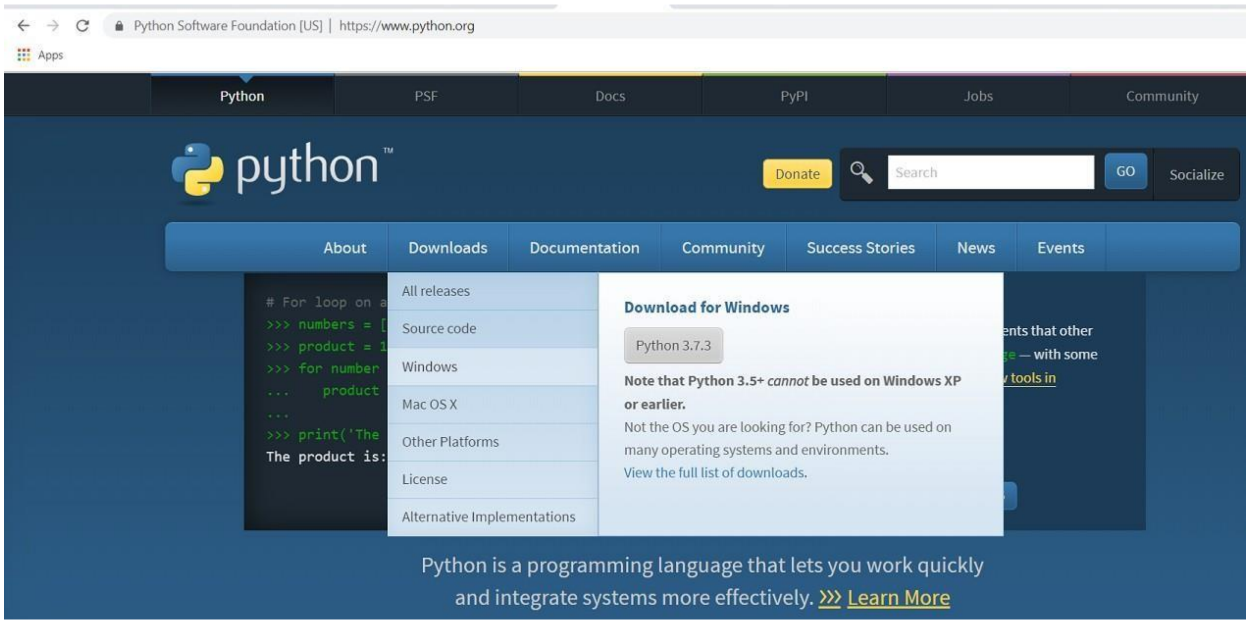
· Python is available on a wide variety of platforms including Linux and Mac OS X. Let'sunderstand how to set up our Python environment.
· The most up-to-date and current source code, binaries, documentation, news, etc., is availableon the official website of Python.

2.2.1 Installation (using python IDLE):

· Installing python is generally easy, and nowadays many Linux and Mac OS distributionsinclude a recent python.
· Download python from www.python.org

· When the download is completed, double click the file and follow the instructions to install it.

· When python is installed, a program called IDLE is also installed along with it. It provides a graphical user interface to work with python.



```
# For loop on a
>>> numbers = [
>>> product = 1
>>> for number
...     product
...
>>> print('The
The product is:
```

Note that Python 3.5+ *cannot* be used on Windows XP or earlier.

Not the OS you are looking for? Python can be used on many operating systems and environments.

View the full list of downloads.

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

Fig 2.2.1: Python download

2.2.2 Installation (using Anaconda):

· Python programs are also executed using Anaconda[2].

· Anaconda is a free open-source distribution of python for large scale data processing, predictive analytics and scientific computing.

· Conda is a package manager that quickly installs and manages packages.

· In WINDOWS:

· Step 1: Open Anaconda.com/downloads in the web browser.

· Step 2: Download python 3.4 version for (32-bits graphic installer/64 -bit graphic installer)

· Step 3: select installation type (all users)

· Step 4: Select path (i.e. add anaconda to path & register anaconda as default python 3.4) nextclick install and next click finish

· Step 5: Open Jupiter notebook (it opens in default browser)

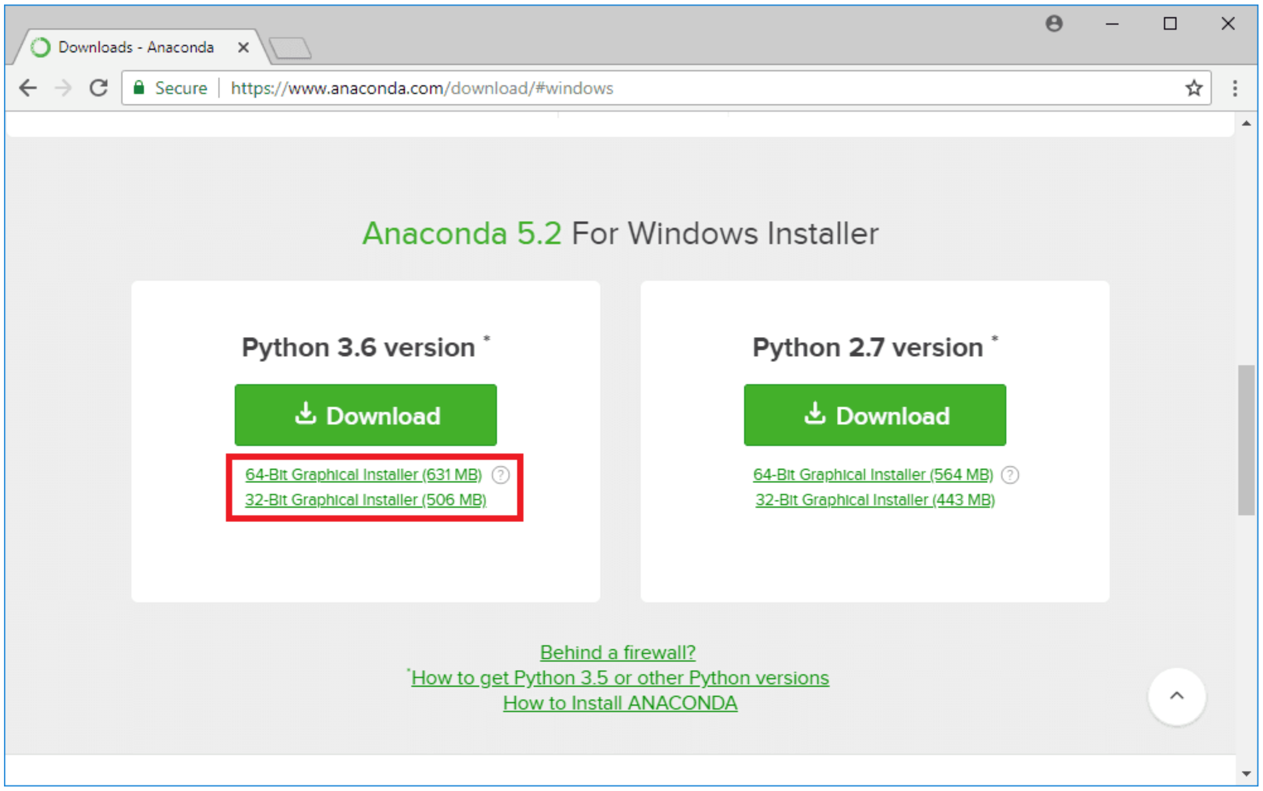

Downloads - Anaconda
Secure
https://www.anaconda.com/download/#windows
Anaconda 5.2 For Windows Installer
Python 3.6 version *
Download
64-Bit Graphical Installer (631 MB)
32-Bit Graphical Installer (506 MB)
Python 2.7 version *
Download
64-Bit Graphical Installer (564 MB)
32-Bit Graphical Installer (443 MB)
Behind a firewall?
*How to get Python 3.5 or other Python versions
How to Install ANACONDA

Fig 2.2.2: Anaconda download

2.3 FEATURES OF PYTHON:

· Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntax. Thisallows the 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-maintaining.

· A broad standard library: Python's bulk of the library is very portable and cross platformcompatible on UNIX, Windows, and Macintosh[2].

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

· Extendable: You can add low-level modules to the Python interpreter.

·These modules enable 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.

2.4 PYTHON 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. Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore,by assigning different data types to variables, you can store integers, decimals or characters in these variables.
Python has five standard data types are:
· Numbers
· Strings
· Lists
· Tuples
· Dictionary

2.4.1 Python Numbers:

· Number data types store numeric values. Number objects are created when you assign a valueto them.
· Python supports four different numerical types − int (signed integers) long (long integers, theycan also be represented in octal and hexadecimal) float (floating point real values) complex (complex numbers).

2.4.2 Python Strings:

· Strings in Python are identified as a contiguous set of characters represented in the quotationmarks.
· Python allows for either pairs of single or double quotes.
· Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 inthe beginning of the string and working their way from -1 at the end.
· The plus (+) sign is the string concatenation operator and the asterisk (*) is the repetitionoperator.

2.4.3 Python Lists:

· Lists are the most versatile of Python's compound data types.
· A list contains items separated by commas and enclosed within square brackets ([]).
· To some extent, lists are similar to arrays in C. One difference between them is that all theitems belonging to a list can be of different data types.
· The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1. · The plus (+) sign isthe list concatenation operator, and the asterisk (*) is the repetition operator.

2.4.4 Python Tuples:

· A tuple is another sequence data type that is similar to the list.
· A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.
· The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated.
· Tuples can be thought of as read-only lists.
· For example − Tuples are fixed size in nature whereas lists are dynamic. In other words, a tupleis immutable whereas a list is mutable. You can't add elements to a tuple.

· Tuples have no append or extend method. You can't remove elements from a tuple. Tupleshave no remove or pop method.

2.4.5 Python Dictionary:

· Python's dictionaries are a kind of hash table type.
· They work like associative arrays or hashes found in Perl and consist of key-value pairs. · A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.
· Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).
· You can use numbers to "index" into a list, meaning you can use numbers to find out what's inlists. You should know this about lists by now, but make sure you understand that you can only use numbers to get items out of a list.
· What a dict does is let you use anything, not just numbers. Yes, a dict associates one thing toanother, no matter what it is.

2.5 PYTHON FUNCTION:

2.5.1 Defining a Function:

You can define functions to provide the required functionality[2]. Here are simple rules to define a function in Python. Function blocks begin with the keyword def followed by the function nameand parentheses (i.e.()).

Any input parameters or arguments should be placed within these parentheses. You can alsodefine parameters inside these parentheses.

The code block within every function starts with a colon (:) and is indented. The statement returns [expression] exits a function, optionally passing back an expression to the caller. A returnstatement with no arguments is the same as return None.

2.5.2 Calling a Function:

Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code. Once the basic structure of a function is finalized, youcan execute it by calling it from another function or directly from the Python prompt.

2.6 PYTHON USING OOPs CONCEPTS:

2.6.1 Class:

A user-defined prototype for an object that defines a set of attributes that characterize any objectof the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.
· Class variable: A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently asinstance variables are.
· Data member: A class variable or instance variable that holds data associated with a class andits objects.
· Instance variable: A variable that is defined inside a method and belongs only to the currentinstance of a class.
· Defining a Class:
· We define a class in a very similar way to how we define a function.
· Just like a function ,we use parentheses and a colon after the class name(i.e. ():). Fig 3.6.1.1:Defining a Class

2.6.2 __init__method in Class:

● The init method — also called a constructor — is a special method that runs when an instanceis created so we can perform any tasks to set up the instance.

● The init method has a special name that starts and ends with two underscores:_init_().

CHAPTER 3: ENSEMBLE METHODS

3.1.Introduction:

Ensemble methods is a machine learning technique that combines several base models in order to produce one optimal predictive model. To better understand this definition lets take a step back into ultimate goal of machine learning and model building. This is going to make more sense as I dive into specific examples and why Ensemble methods are used[3].

3.1.1. Averaging Method

It is mainly used for regression problems. The method consists of building multiple models independently and returning the average of the prediction of all the models. In general, the combined output is better than an individual output because variance is reduced[3].

3.1.2. Max Voting

It is mainly used for classification problems. The method consists of building multiple models independently and getting their individual output called ‘vote’. The class with maximum votes is returned as output[3].

3.1.3. Bagging

It is also known as a bootstrapping method. Base models are run on bags to get a fair distribution of the whole dataset. A bag is a subset of the dataset along with a replacement to make the size of the bag the same as the whole dataset. The final output is formed after combining the output of all base models[3].

3.1.4. Boosting

Boosting is a sequential method–it aims to prevent a wrong base model from affecting the final output. Instead of combining the base models, the method focuses on building a new model that is dependent on the previous one. A new model tries to remove the errors made by its previous one. Each of these models is called weak learners. The final model (aka strong learner) is formed by getting the weighted mean of all the weak learners[3].

3.1.5. KNORA-E

This method searches for a local Oracle, which is a base classifier that correctly classify all samples belonging to the region of competence of the test sample. All classifiers with a perfect performance in the region of competence are selected (local Oracles). In the case that no classifier achieves a perfect accuracy, the size of the competence region is reduced (by removing the farthest neighbor) and the performance of the classifiers are re-evaluated. The outputs of the selected ensemble of classifiers is combined using the majority voting scheme. If no base classifier is selected, the whole pool is used for classification[4].

CHAPTER 4: TWITTER SENTIMENT ANALYSIS

4.1.LITERATURE REVIEW:

Twitter sentiment analysis has gained significant attention in recent years due to the explosive growth of social media platforms. This review focuses on the domain of airline passenger services, particularly emphasizing the research conducted based on the foundational paper titled "Airline passenger's sentiment analysis for improving the quality of airline services by using a deep learning approach." The paper serves as a cornerstone for exploring the diverse methodologies, techniques, and advancements in Twitter sentiment analysis concerning airline services.

4.1.1.Social Media and Sentiment Analysis:

Social media platforms, especially Twitter, have become rich sources of real-time data reflecting people's opinions and sentiments. Researchers have extensively explored sentiment analysis techniques to extract valuable insights from this data, enhancing our understanding of customer experiences and preferences in various domains, including the airline industry.

4.1.2. Deep Learning Approaches:

The base paper's focus on utilizing deep learning methods for sentiment analysis has spurred further research in this area. Deep learning algorithms, such as Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks, have shown remarkable performance in capturing complex patterns within textual data. Researchers have built upon this foundation, experimenting with different architectures and hybrid models to improve accuracy and efficiency in sentiment classification tasks specific to airline passenger services.

4.1.3. Feature Engineering and Text Representation:

Feature engineering and text representation techniques play a crucial role in sentiment analysis. Researchers have explored various methods, including word embeddings (e.g., Word2Vec, GloVe), n-grams, and syntactic features, to enhance the representation of textual data. These techniques enable the extraction of meaningful features, aiding sentiment classifiers in understanding the context and nuances of tweets related to airline services.

4.1.4. Aspect-Based Sentiment Analysis:

Understanding sentiment at a finer granularity, such as aspect-based sentiment analysis, has gained prominence. Researchers have extended the analysis from overall sentiment to specific aspects of airline services, such as flight punctuality, customer service, and in-flight amenities. This nuanced approach provides airlines with detailed feedback, facilitating targeted improvements in areas that directly impact passenger satisfaction.

4.1.5. Real-Time Sentiment Analysis for Airline Services:

Real-time sentiment analysis is essential for airlines to promptly respond to customer feedback and mitigate potential issues. Researchers have explored techniques for efficient real-time sentiment analysis, integrating natural language processing tools with streaming data processing frameworks. These studies have contributed to the development of systems capable of monitoring and analyzing tweets in real-time, enabling airlines to enhance their services on-the-fly.

4.1.6. Challenges and Ethical Considerations:

While sentiment analysis has shown great promise, researchers have also identified challenges and ethical considerations. Issues related to data privacy, bias in training data, and the interpretability of deep learning models have been subjects of active research. Addressing these challenges is crucial to ensuring the reliability and fairness of sentiment analysis results, especially in sensitive domains like airline passenger services.

4.2. PROJECT REQUIREMENTS:

4.2.1. Packages used:

Numpy: In Python we have lists that serve the purpose of arrays, but they are slow to process. Numpy aims to provide an array object that is up to 50x faster than traditional Python lists. Thearray object in Numpy is called ndata, it provides a lot of supporting functions that make working with ndarray very easy. Arrays are very frequently used in data science, where speed and resources are very important[5].

Pandas: Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data. Python with Pandas isused in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc[5].

Scikit-learn: also called Sklearn, is a robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling, including classification, regression, clustering, and dimensionality reduction via a consistent interface[5].

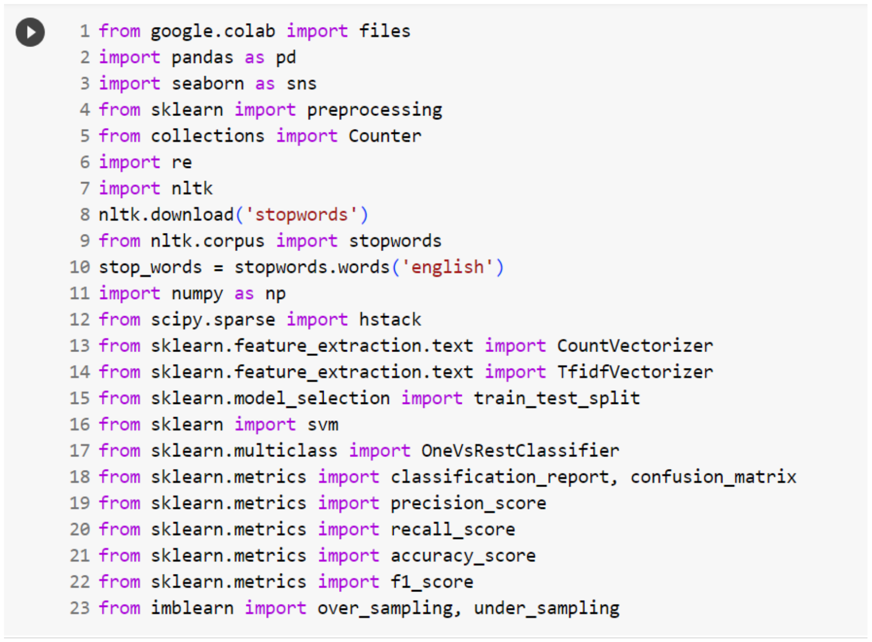
Collections: The collection Module in Python provides different types of containers. A Container is an object that is used to store different objects and provide a way to access the contained objects and iterate over them. Some of the built-in containers are Tuple, List, Dictionary, etc[7].

Requests: The collection Module in Python provides different types of containers. A Container is an object that is used to store different objects and provide a way to access the contained objects and iterate over them. Some of the built-in containers are Tuple, List, Dictionary, etc[8].

NLTK: The Natural Language Toolkit (NLTK) is a platform used for building Python programs that work with human language data for applying in statistical natural language processing (NLP). It contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning. It also includes graphical demonstrations and sample data sets as well as accompanied by a cook book and a book which explains the principles behind the underlying language processing tasks that NLTK supports[9].

Imblearn: Imbalanced-Learn is a Python module that helps in balancing the datasets which are highly skewed or biased towards some classes. Thus, it helps in resampling the classes which are otherwise oversampled or undesampled. If there is a greater imbalance ratio, the output is biased to the class which has a higher number of examples[6].

Seaborn: is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas' data structures. Seaborn helps you explore and understand your data.



```
from google.colab import files
import pandas as pd
import seaborn as sns
from sklearn import preprocessing
from collections import Counter
import re
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
import numpy as np
from scipy.sparse import hstack
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.multiclass import OneVsRestClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from imblearn import over_sampling, under_sampling
```

Fig 4.1.1: Packages used

4.2.2. Algorithms Used

· Random Forest

· XGBoost

· Linear Regression

· Averaging Method

· Max Voting

· Bagging

· Boosting

· KNORA-E

4.3 PROBLEM STATEMENT:

The aim of this dataset is to predict the sentiment of the passenger or traveller of the airlines through their comments in twitter

4.4 DATASET DESCRIPTION:

The dataset is also distributed as a CSV formatted file hearts.csv. The loaded object contains ‘tweet_id’, ‘airline_sentiment’, ‘airline_sentiment_confidence’, ‘negativereason’, ‘negativereason_confidence’, ‘airline’, ‘airline_sentiment_gold’, ‘name’, ‘negativereason_gold’, ‘retweet_count’, ‘text’, ‘tweet_coord’, ‘tweet_created’, ‘tweet_location’, ‘user_timezone’.

4.5 OBJECTIVE OF THE CASE STUDY:

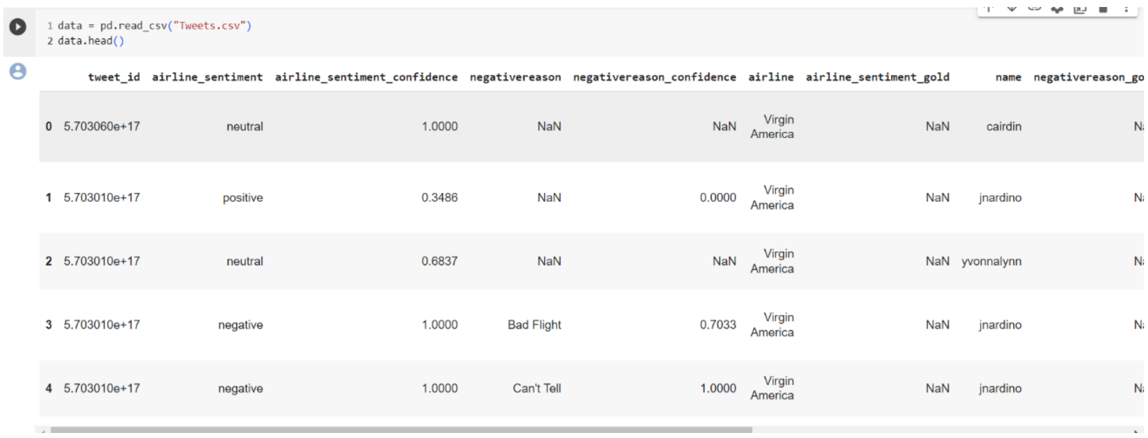
Objective of the problem is to analyze Twitter sentiment related to airlines with the goal of understanding customer perceptions, identifying prevalent issues, and gauging overall customer satisfaction. By employing sentiment analysis techniques, this study aims to categorize tweets into positive, negative, or neutral sentiments, enabling airlines to proactively address concerns, enhance customer experience, and optimize their services based on real-time feedback.

CHAPTER 5: DATA PREPROCESSING/ FEATUREENGINEERING AND EDA

5.1 LOADING THE DATA:

Python has a built-in head () function to read a file. This function returns a file object, also calleda handle, as it is used to read or modify the file accordingly.

We can specify the mode while opening a file. In mode, we specify whether we want to read, write or append to the file. We can also specify if we want to open the file in text mode or binarymode.

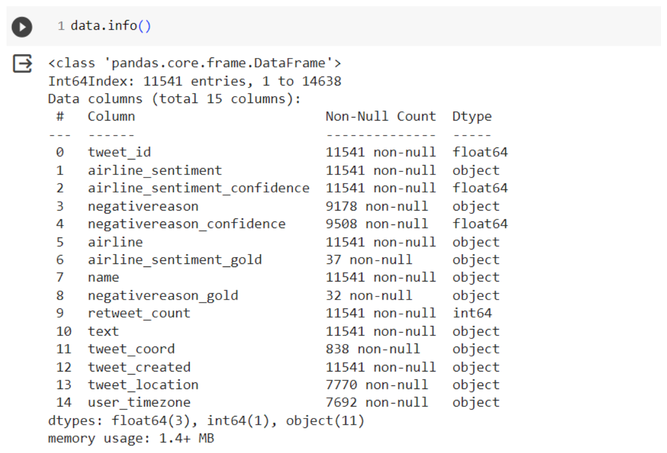


```
data = pd.read_csv("Tweets.csv")
data.head()
```

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	airline	airline_sentiment_gold	name	negativereason_go
0	5.703060e+17	neutral	1.0000	NaN	NaN	Virgin America	NaN	cairdin	N
1	5.703010e+17	positive	0.3486	NaN	0.0000	Virgin America	NaN	jnardino	N
2	5.703010e+17	neutral	0.6837	NaN	NaN	Virgin America	NaN	yvonnalynn	N
3	5.703010e+17	negative	1.0000	Bad Flight	0.7033	Virgin America	NaN	jnardino	N
4	5.703010e+17	negative	1.0000	Can't Tell	1.0000	Virgin America	NaN	jnardino	N

Fig 5.1.1: Loading data

Pandas DataFrame to which all the operations can be performed which helps us to access each and every row as well as columns and each and every value can be accessed using the dataframe.Any missing value or NaN value have to be cleaned.



```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 11541 entries, 1 to 14638
Data columns (total 15 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   tweet_id                      11541 non-null  float64
 1   airline_sentiment             11541 non-null  object 
 2   airline_sentiment_confidence  11541 non-null  float64
 3   negativereason                9178 non-null   object 
 4   negativereason_confidence     9508 non-null   float64
 5   airline                       11541 non-null  object 
 6   airline_sentiment_gold        37 non-null     object 
 7   name                          11541 non-null  object 
 8   negativereason_gold           32 non-null     object 
 9   retweet_count                 11541 non-null  int64  
 10  text                          11541 non-null  object 
 11  tweet_coord                   838 non-null    object 
 12  tweet_created                 11541 non-null  object 
 13  tweet_location                7770 non-null   object 
 14  user_timezone                 7692 non-null   object 
dtypes: float64(3), int64(1), object(11)
memory usage: 1.4+ MB
```

Fig 5.1.2: Reading Data

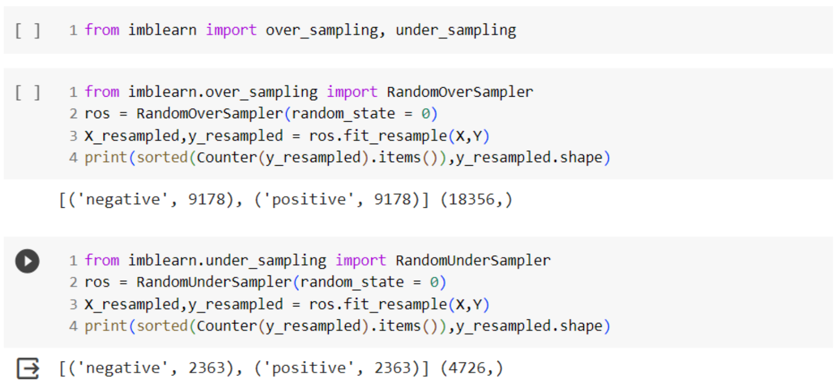
5.2 BALANCING THE DATASET:

Imbalanced data means the number of rows or frequency of data points of one class is much more than the other class. In other words, the ratio of the value counts of classes is much higher. Such data set is known as an imbalanced dataset in which the class having more data points is the majority class and the other is the minority class.

The imbalance makes the classification more challenging. Whenever we build a classifier with such data, it works well with the majority class but gives a poor performance with the minority class. Although, the model performance concerning the minority class matters the most. Some Machine Learning algorithms are more sensitive toward imbalanced data, such as Logistic Regression and Support Vector Machine. However, some algorithms tackle this issue themselves, such as Random Forest and XGBoost[6].

There are two sampling techniques available to handle the imbalanced data:

1. Under Sampling
2. Over Sampling



```
from imblearn import over_sampling, under_sampling
```

```
from imblearn.over_sampling import RandomOverSampler
ros = RandomOverSampler(random_state = 0)
X_resampled,y_resampled = ros.fit_resample(X,Y)
print(sorted(Counter(y_resampled).items()),y_resampled.shape)
```

```
[('negative', 9178), ('positive', 9178)] (18356,)
```

```
from imblearn.under_sampling import RandomUnderSampler
ros = RandomUnderSampler(random_state = 0)
X_resampled,y_resampled = ros.fit_resample(X,Y)
print(sorted(Counter(y_resampled).items()),y_resampled.shape)
```

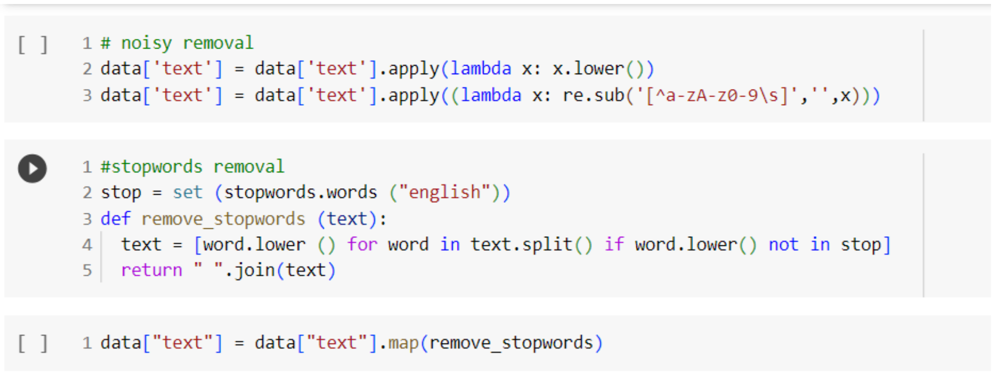
```
[('negative', 2363), ('positive', 2363)] (4726,)
```

Fig 5.2.1: Balancing the Dataset

5.3 NOISE AND STOPWORD REMOVAL:

Noise removal in the context of text data refers to the process of eliminating irrelevant or unnecessary information that does not contribute to the analysis or understanding of the text. In the context of Twitter sentiment analysis, noise can include special characters, numbers, URLs, and other elements that do not carry significant meaning regarding sentiment.

Stopword removal is a common step in text preprocessing for natural language processing tasks, including sentiment analysis. Stopwords are words that are considered to be of little value in terms of content analysis because they are very common and do not carry significant meaning. Examples of stopwords in English include "and," "is," "the," "in," etc. Removing stopwords can reduce the dimensionality of the data and improve the efficiency of algorithms, especially in tasks like sentiment analysis where the sentiment-carrying words are often adjectives, verbs, or specific nouns[11].



```
# noisy removal
data['text'] = data['text'].apply(lambda x: x.lower())
data['text'] = data['text'].apply((lambda x: re.sub('[^a-zA-z0-9\s]','',x)))
```

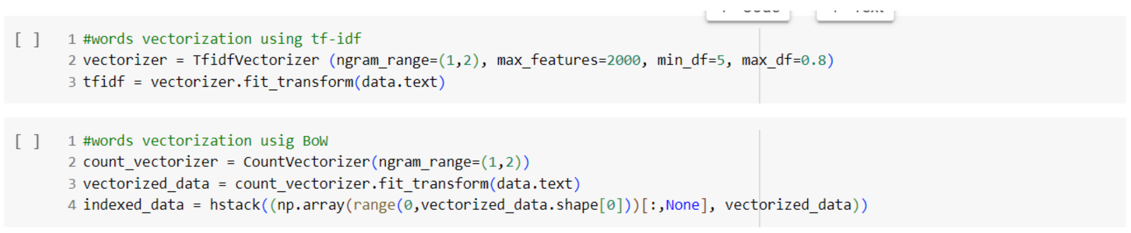
```
#stopwords removal
stop = set (stopwords.words ("english"))
def remove_stopwords (text):
  text = [word.lower () for word in text.split() if word.lower() not in stop]
  return " ".join(text)
```

```
data["text"] = data["text"].map(remove_stopwords)
```

Fig 5.3.1: Noise and Stopword removal

5.4 WORD VECTORIZATION:

Word vectorization techniques like Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF) are essential in natural language processing tasks, including sentiment analysis. They transform text data into numerical vectors that machine learning algorithms can understand[12].



```
[ ]  #words vectorization using tf-idf
     vectorizer = TfidfVectorizer (ngram_range=(1,2), max_features=2000, min_df=5, max_df=0.8)
     tfidf = vectorizer.fit_transform(data.text)
```

```
[ ]  #words vectorization usig BoW
     count_vectorizer = CountVectorizer(ngram_range=(1,2))
     vectorized_data = count_vectorizer.fit_transform(data.text)
     indexed_data = hstack((np.array(range(0,vectorized_data.shape[0]))[:,None], vectorized_data))
```

Fig 5.4.1: Word vectorization

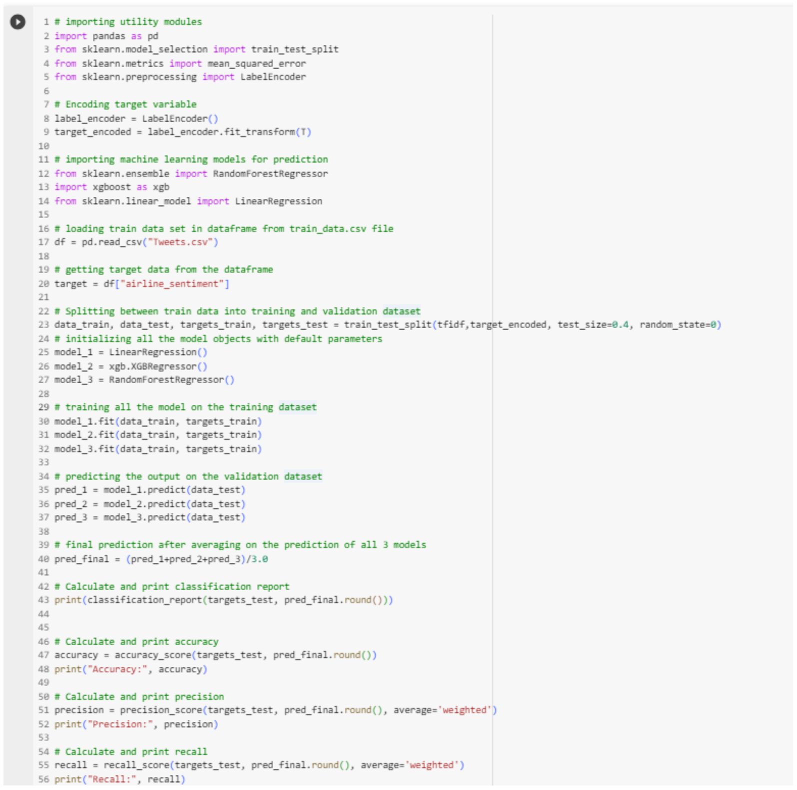
CHAPTER 6: MODEL BUILDING AND EVALUATION

6.1. BRIEF ABOUT THE ALGORITHMS USED AND BUILDING MODEL

We have used ensemble methods to predict whether the data in twitter about the airlines review given is accurate or not. We have used five different models to check the accuracy and all evaluation metrics for the given dataset. We will be explaining the methods below:

1.Averaging method:

It is mainly used for regression problems. The method consists of building multiple models independently and returning the average of the prediction of all the models. In general, the combined output is better than an individual output because variance is reduced. Here we used three methods, then we will average the predictions of all models[3].

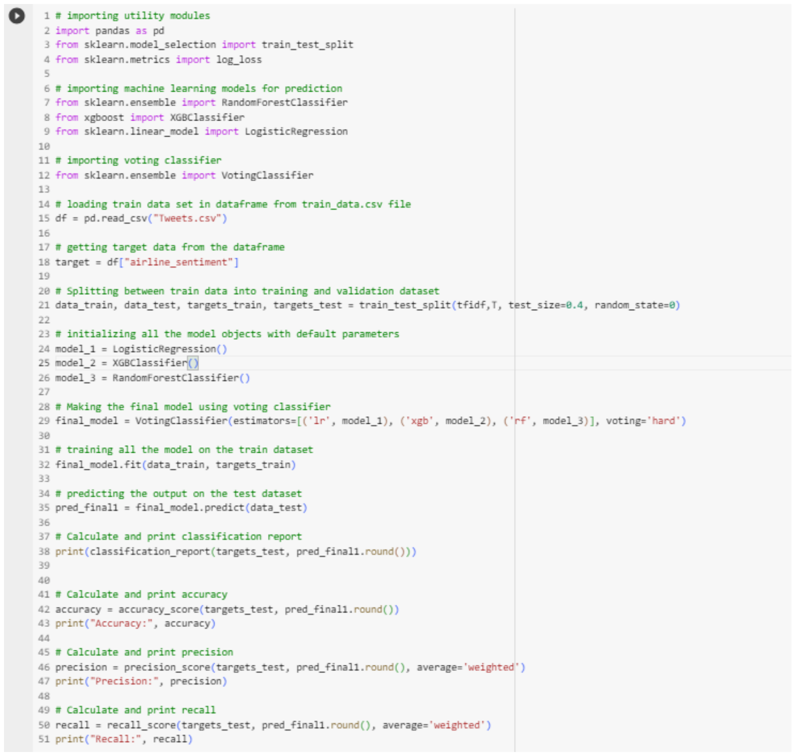


```python
# importing utility modules
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import LabelEncoder

# Encoding target variable
label_encoder = LabelEncoder()
target_encoded = label_encoder.fit_transform(T)

# importing machine learning models for prediction
from sklearn.ensemble import RandomForestRegressor
import xgboost as xgb
from sklearn.linear_model import LinearRegression

# loading train data set in dataframe from train_data.csv file
df = pd.read_csv("Tweets.csv")

# getting target data from the dataframe
target = df["airline_sentiment"]

# Splitting between train data into training and validation dataset
data_train, data_test, targets_train, targets_test = train_test_split(tfidf,target_encoded, test_size=0.4, random_state=0)
# initializing all the model objects with default parameters
model_1 = LinearRegression()
model_2 = xgb.XGBRegressor()
model_3 = RandomForestRegressor()

# training all the model on the training dataset
model_1.fit(data_train, targets_train)
model_2.fit(data_train, targets_train)
model_3.fit(data_train, targets_train)

# predicting the output on the validation dataset
pred_1 = model_1.predict(data_test)
pred_2 = model_2.predict(data_test)
pred_3 = model_3.predict(data_test)

# final prediction after averaging on the prediction of all 3 models
pred_final = (pred_1+pred_2+pred_3)/3.0

# Calculate and print classification report
print(classification_report(targets_test, pred_final.round()))


# Calculate and print accuracy
accuracy = accuracy_score(targets_test, pred_final.round())
print("Accuracy:", accuracy)

# Calculate and print precision
precision = precision_score(targets_test, pred_final.round(), average='weighted')
print("Precision:", precision)

# Calculate and print recall
recall = recall_score(targets_test, pred_final.round(), average='weighted')
print("Recall:", recall)
```

Fig 6.1.1: Averaging Method

2.Max Voting:

It is mainly used for classification problems. The method consists of building multiple models independently and getting their individual output called 'vote'. The class with maximum votes is returned as output[3].

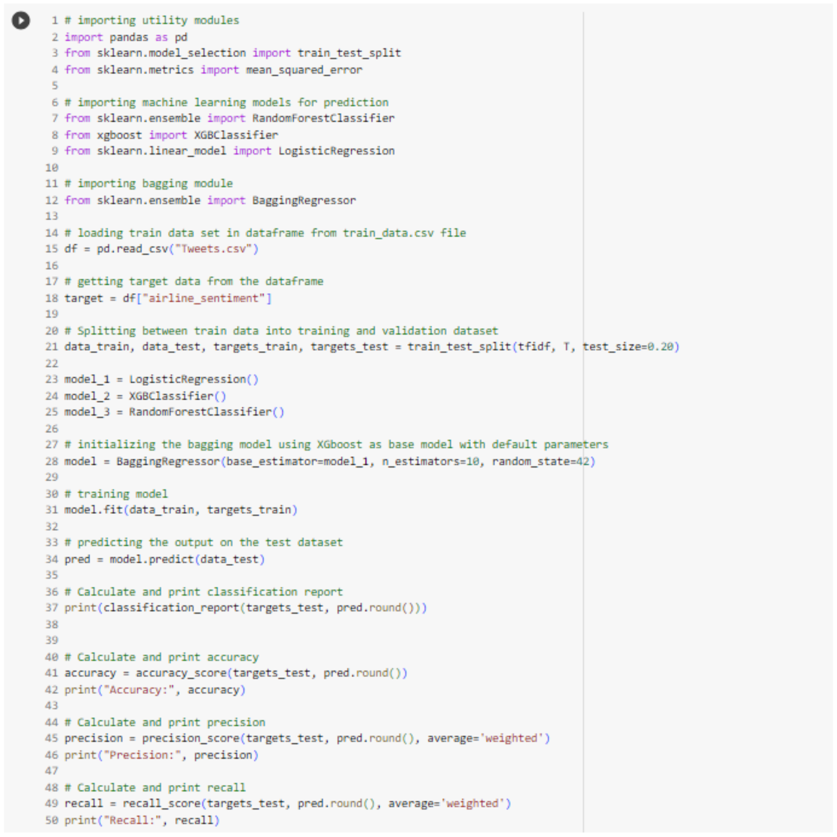


```python
# importing utility modules
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import log_loss

# importing machine learning models for prediction
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression

# importing voting classifier
from sklearn.ensemble import VotingClassifier

# loading train data set in dataframe from train_data.csv file
df = pd.read_csv("Tweets.csv")

# getting target data from the dataframe
target = df["airline_sentiment"]

# Splitting between train data into training and validation dataset
data_train, data_test, targets_train, targets_test = train_test_split(tfidf,T, test_size=0.4, random_state=0)

# initializing all the model objects with default parameters
model_1 = LogisticRegression()
model_2 = XGBClassifier()
model_3 = RandomForestClassifier()

# Making the final model using voting classifier
final_model = VotingClassifier(estimators=[('lr', model_1), ('xgb', model_2), ('rf', model_3)], voting='hard')

# training all the model on the train dataset
final_model.fit(data_train, targets_train)

# predicting the output on the test dataset
pred_final1 = final_model.predict(data_test)

# Calculate and print classification report
print(classification_report(targets_test, pred_final1.round()))


# Calculate and print accuracy
accuracy = accuracy_score(targets_test, pred_final1.round())
print("Accuracy:", accuracy)

# Calculate and print precision
precision = precision_score(targets_test, pred_final1.round(), average='weighted')
print("Precision:", precision)

# Calculate and print recall
recall = recall_score(targets_test, pred_final1.round(), average='weighted')
print("Recall:", recall)
```

Fig 6.1.2: Max Voting

3. Bagging:

It is also known as a bootstrapping method. Base models are run on bags to get a fair distribution of the whole dataset. A bag is a subset of the dataset along with a replacement to make the size of the bag the same as the whole dataset. The final output is formed after combining the output of all base models[3].

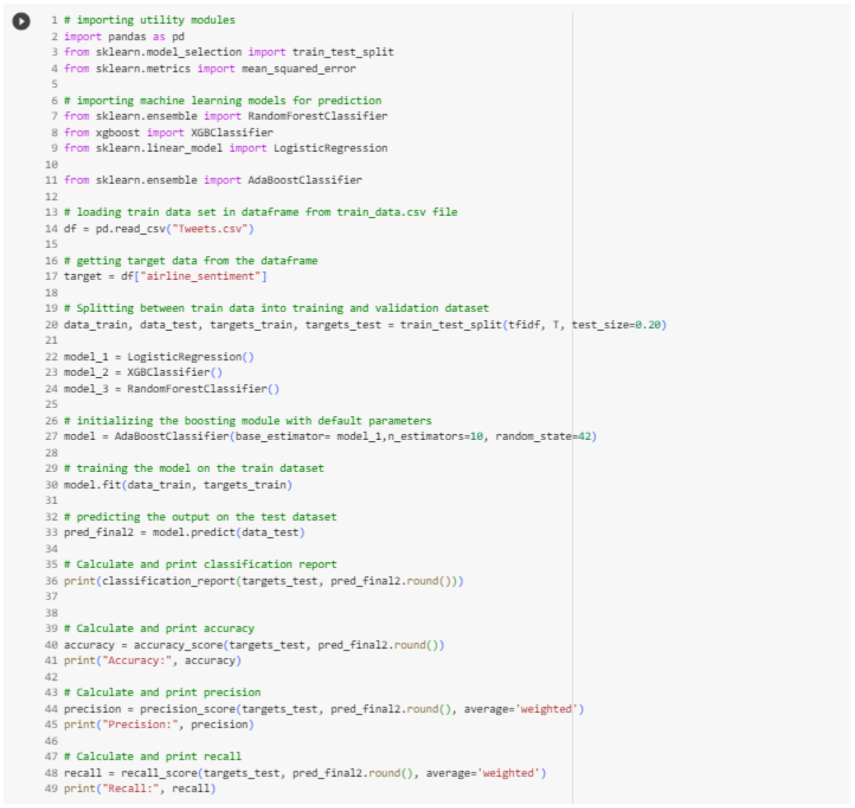


```python
# importing utility modules
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# importing machine learning models for prediction
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression

# importing bagging module
from sklearn.ensemble import BaggingRegressor

# loading train data set in dataframe from train_data.csv file
df = pd.read_csv("Tweets.csv")

# getting target data from the dataframe
target = df["airline_sentiment"]

# Splitting between train data into training and validation dataset
data_train, data_test, targets_train, targets_test = train_test_split(tfidf, T, test_size=0.20)

model_1 = LogisticRegression()
model_2 = XGBClassifier()
model_3 = RandomForestClassifier()

# initializing the bagging model using XGboost as base model with default parameters
model = BaggingRegressor(base_estimator=model_1, n_estimators=10, random_state=42)

# training model
model.fit(data_train, targets_train)

# predicting the output on the test dataset
pred = model.predict(data_test)

# Calculate and print classification report
print(classification_report(targets_test, pred.round()))


# Calculate and print accuracy
accuracy = accuracy_score(targets_test, pred.round())
print("Accuracy:", accuracy)

# Calculate and print precision
precision = precision_score(targets_test, pred.round(), average='weighted')
print("Precision:", precision)

# Calculate and print recall
recall = recall_score(targets_test, pred.round(), average='weighted')
print("Recall:", recall)
```

Fig 6.1.3: Bagging

4.Boosting:

Boosting is a sequential method–it aims to prevent a wrong base model from affecting the final output. Instead of combining the base models, the method focuses on building a new model that is dependent on the previous one. A new model tries to remove the errors made by its previous one. Each of these models is called weak learners. The final model (aka strong learner) is formed by getting the weighted mean of all the weak learners[3].

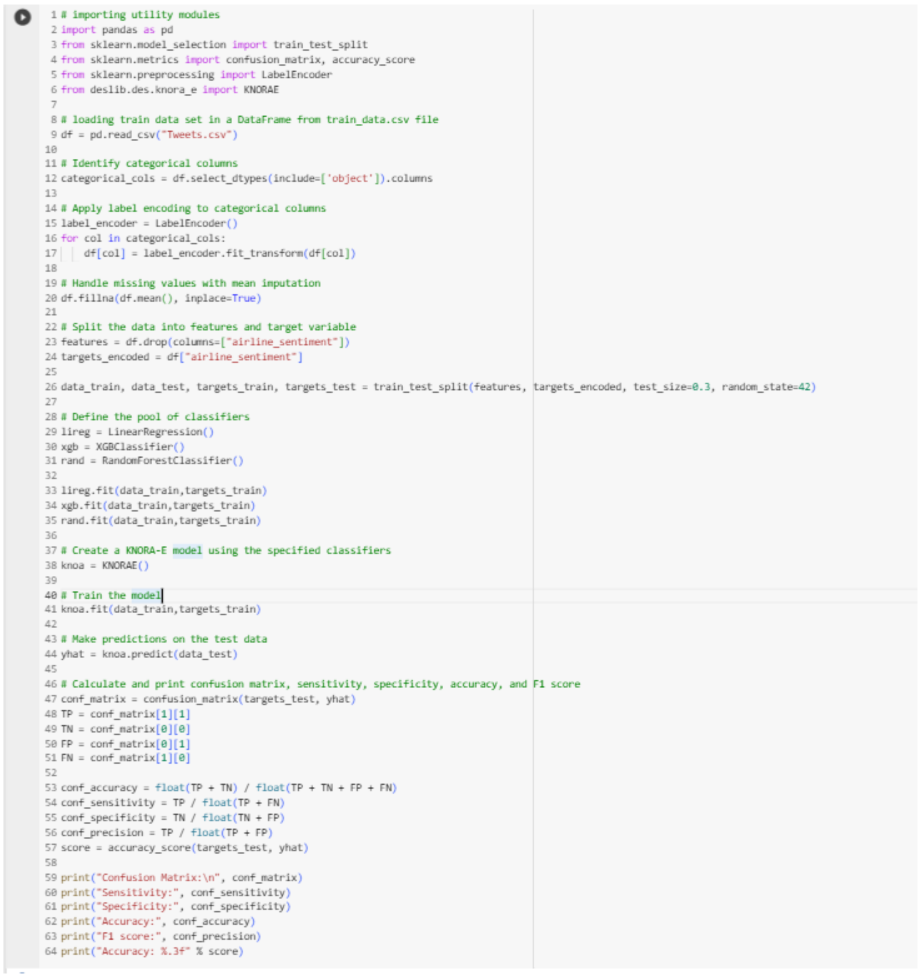


```python
# importing utility modules
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# importing machine learning models for prediction
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression

from sklearn.ensemble import AdaBoostClassifier

# loading train data set in dataframe from train_data.csv file
df = pd.read_csv("Tweets.csv")

# getting target data from the dataframe
target = df["airline_sentiment"]

# Splitting between train data into training and validation dataset
data_train, data_test, targets_train, targets_test = train_test_split(tfidf, T, test_size=0.20)

model_1 = LogisticRegression()
model_2 = XGBClassifier()
model_3 = RandomForestClassifier()

# initializing the boosting module with default parameters
model = AdaBoostClassifier(base_estimator= model_1,n_estimators=10, random_state=42)

# training the model on the train dataset
model.fit(data_train, targets_train)

# predicting the output on the test dataset
pred_final2 = model.predict(data_test)

# Calculate and print classification report
print(classification_report(targets_test, pred_final2.round()))


# Calculate and print accuracy
accuracy = accuracy_score(targets_test, pred_final2.round())
print("Accuracy:", accuracy)

# Calculate and print precision
precision = precision_score(targets_test, pred_final2.round(), average='weighted')
print("Precision:", precision)

# Calculate and print recall
recall = recall_score(targets_test, pred_final2.round(), average='weighted')
print("Recall:", recall)
```

Fig 6.1.4: Boosting

5.KNORA-E:

This method searches for a local Oracle, which is a base classifier that correctly classify all samples belonging to the region of competence of the test sample. All classifiers with a perfect performance in the region of competence are selected (local Oracles). In the case that no classifier achieves a perfect accuracy, the size of the competence region is reduced (by removing the farthest neighbor) and the performance of the classifiers are re-evaluated. The outputs of the selected ensemble of classifiers is combined using the majority voting scheme. If no base classifier is selected, the whole pool is used for classification[4].

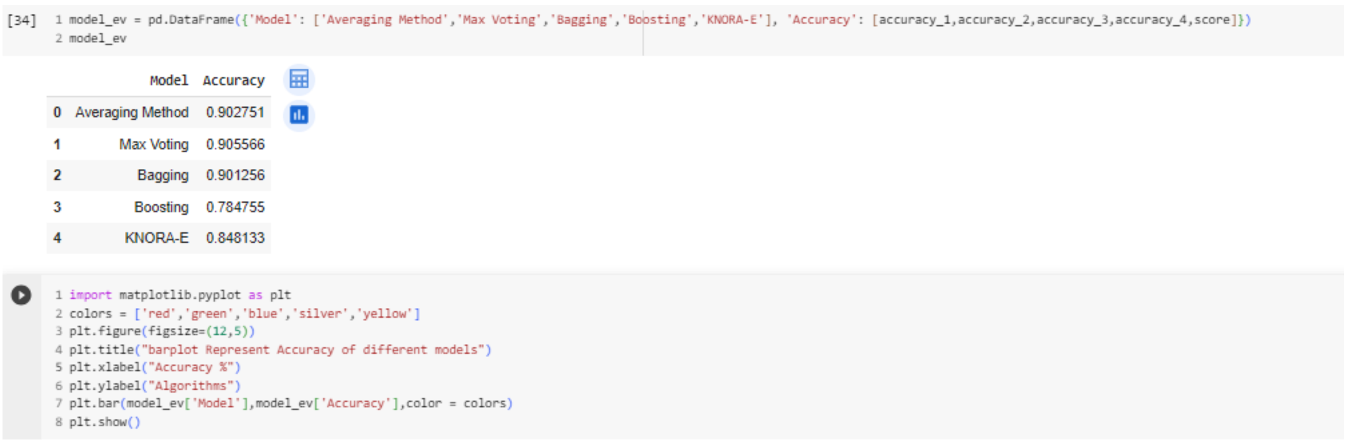


```python
# importing utility modules
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.preprocessing import LabelEncoder
from deslib.des.knora_e import KNORAE

# loading train data set in a DataFrame from train_data.csv file
df = pd.read_csv("Tweets.csv")

# Identify categorical columns
categorical_cols = df.select_dtypes(include=['object']).columns

# Apply label encoding to categorical columns
label_encoder = LabelEncoder()
for col in categorical_cols:
    df[col] = label_encoder.fit_transform(df[col])

# Handle missing values with mean imputation
df.fillna(df.mean(), inplace=True)

# Split the data into features and target variable
features = df.drop(columns=["airline_sentiment"])
targets_encoded = df["airline_sentiment"]

data_train, data_test, targets_train, targets_test = train_test_split(features, targets_encoded, test_size=0.3, random_state=42)

# Define the pool of classifiers
lireg = LinearRegression()
xgb = XGBClassifier()
rand = RandomForestClassifier()

lireg.fit(data_train,targets_train)
xgb.fit(data_train,targets_train)
rand.fit(data_train,targets_train)

# Create a KNORA-E model using the specified classifiers
knoa = KNORAE()

# Train the model
knoa.fit(data_train,targets_train)

# Make predictions on the test data
yhat = knoa.predict(data_test)

# Calculate and print confusion matrix, sensitivity, specificity, accuracy, and F1 score
conf_matrix = confusion_matrix(targets_test, yhat)
TP = conf_matrix[1][1]
TN = conf_matrix[0][0]
FP = conf_matrix[0][1]
FN = conf_matrix[1][0]

conf_accuracy = float(TP + TN) / float(TP + TN + FP + FN)
conf_sensitivity = TP / float(TP + FN)
conf_specificity = TN / float(TN + FP)
conf_precision = TP / float(TP + FP)
score = accuracy_score(targets_test, yhat)

print("Confusion Matrix:\n", conf_matrix)
print("Sensitivity:", conf_sensitivity)
print("Specificity:", conf_specificity)
print("Accuracy:", conf_accuracy)
print("F1 score:", conf_precision)
print("Accuracy: %.3f" % score)
```

Fig 6.1.5: KNORA-E

6.3. MODEL EVALUATION

Model Evaluation can give us a brief information on the accuracies of the machine learningmodels we have used to predict the correct information. In all the models which has high accuracy can give us a correct prediction of heart disease.



```
[34] model_ev = pd.DataFrame({'Model': ['Averaging Method','Max Voting','Bagging','Boosting','KNORA-E'], 'Accuracy': [accuracy_1,accuracy_2,accuracy_3,accuracy_4,score]})
     model_ev
```

	Model	Accuracy
0	Averaging Method	0.902751
1	Max Voting	0.905566
2	Bagging	0.901256
3	Boosting	0.784755
4	KNORA-E	0.848133

```
import matplotlib.pyplot as plt
colors = ['red','green','blue','silver','yellow']
plt.figure(figsize=(12,5))
plt.title("barplot Represent Accuracy of different models")
plt.xlabel("Accuracy %")
plt.ylabel("Algorithms")
plt.bar(model_ev['Model'],model_ev['Accuracy'],color = colors)
plt.show()
```

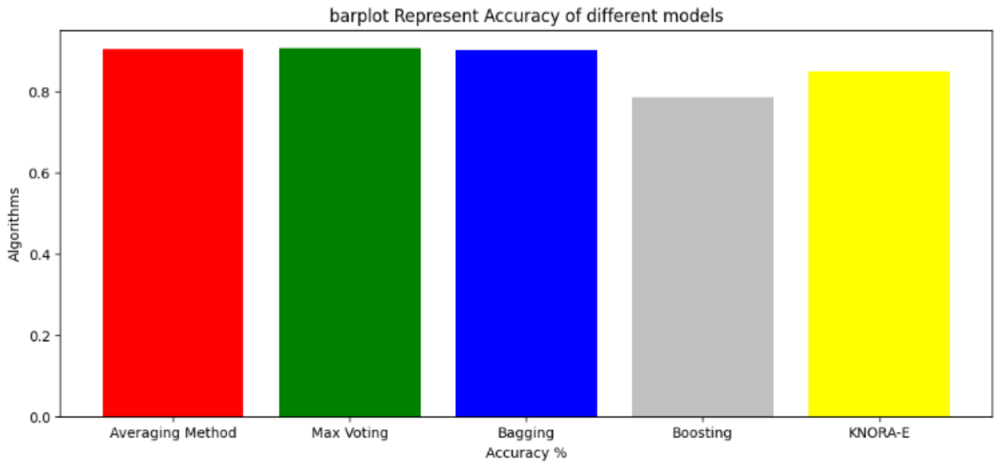


Fig 6.2.1 Model Validation

- Hence, here the accuracy of the Max Voting is high.
- So, it gives us an accurate and reliable prediction on sentiment of the tweets of the airline passengers.

CONCLUSION

Using ensemble techniques, specifically Max Voting, produced remarkably accurate results in our sentiment analysis of airline passengers on Twitter. This cooperative strategy, which combined various models, gave rich insights regarding passenger attitude that are essential for improving airline services. Airlines were able to address customer issues and build closer relationships by leveraging the strength of ensemble approaches, which demonstrated an ability to interpret complicated feelings. This study highlights Max Voting's efficacy and emphasizes the value of continuous analysis in order to adjust to changing consumer emotions in the ever-changing world of social media.

REFERENCES

1. "Machine Learning." Wikipedia, Wikimedia Foundation, 16 Oct. 2023, en.wikipedia.org/wiki/Machine_learning.

2. "Python (Programming Language)." *Wikipedia*, Wikimedia Foundation, 18 Oct. 2023, en.wikipedia.org/wiki/Python_(programming_language).

3. "Ensemble Methods in Python." GeeksforGeeks, GeeksforGeeks, 27 Mar. 2023, www.geeksforgeeks.org/ensemble-methods-in-python/.

4. "K-Nearest Oracle-Eliminate" K-Nearest Oracle-Eliminate (KNORA-E)- Deslib 0.4.Dev Documentation,deslib.readthedocs.io/en/modules/des/knora_e.html:~:text=k%2DNearest%20Or.

5. "Libraries in Python." GeeksforGeeks, GeeksforGeeks, 18 Oct. 2021, www.geeksforgeeks.org/libraries-in-python/.

6. "ML: Handling Imbalanced Data with Smote and near Miss Algorithm in Python." GeeksforGeeks, GeeksforGeeks, 11 Jan. 2023, www.geeksforgeeks.org/ml-handling-imbalanced-data-with-smote-and-near-miss-algorithm-in-python/.

7. "Python Collection Module - Javatpoint." *Www.Javatpoint.Com*, www.javatpoint.com/python-collection-module. Accessed 27 Oct. 2023.

8. "Python's Requests Library (Guide)." *Real Python*, Real Python, 17 Sept. 2022, realpython.com/python-requests/.

9. SudoPurge. "Intro to NLTK for NLP with Python." *Medium*, Towards Data Science, 18 Feb. 2021, towardsdatascience.com/intro-to-nltk-for-nlp-with-python-87da6670dde.

10. Nourbakhsh, A., and M. Rezaei Chelkasari. "Airline passenger’s sentiment analysis for improving the quality of airline services by using a deep learning approach." *International Journal of Applied Operational Research-An Open Access Journal* 11, no. 2 (2023): 77-97.

11. “Python - Remove Stopwords.” *Online Tutorials, Courses, and eBooks Library*, 2015, www.tutorialspoint.com/python_text_processing/python_remove_stopwords.htm.

12. Jha, Abhishek. “Vectorization Techniques in NLP [Guide].” *Neptune.Ai*, 11 Aug. 2023, neptune.ai/blog/vectorization-techniques-in-nlp-guide.