SPAM DETECTION IN SOCIAL MEDIA USING MACHINE LEARNING

ABSTRACT

Buying and selling the goods and services through internet called as electronic network known to be E-commerce. Due to the convenience of e-commerce, the number of users are increased. Meanwhile, the people review of product also increased. In e-commerce websites, fake review is often the major problem. Nowadays, it is known to be common that user can write the review for their purchased product. There are many ways that user can write reviews. Using this opportunity, there is a possibility that spammers can leave fake review. Many users determine the quality of product based on user’s reviews. So, the fake review creates lot of problems on product quality, sales, and economic growth.

The present use of social media has generated incomparable amounts of social data. Since it is cheap and popular communication and information sharing media. Data could be of text, facts or statistics that are accessible by a computer. Nowadays, a big part of people rely on available content in social media in their decisions (e.g. reviews and feedback on a topic or product). This aspect of sharing information to a large number of individuals with ease has attracted social spammers to exploit the network of trust for spreading spam messages to promote personal blogs, advertisements, phishing, scam and so on. Here our aim is to find fake reviews. By detecting fake reviews the accuracy of e-commerce system can be improved. With the increased popularity of online social networks, spammers find these platforms easily accessible to trap users in malicious activities by posting spam messages.

In the work, we have taken Twitter platform and Performed Spam tweet detection. We have evaluated our solution with four machine learning algorithms namely- Random forest classification, logistic regression, XGB Classifier algorithm and LGBM Classifier algorithm. With logistic regression we are able to achieve an accuracy of more than 60%.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Spams are fraud: a criminal activity designed to trick someone out of money or personal details. Methods constantly evolve as scammers look for new ways to commit fraud and avoid detection. Consumers might be contacted by telephone, post, email or even on their doorsteps. In the long history of scams, the internet is a relatively new way for fraudsters to target potential victims and they have been quick to reinvent old tricks for new digital platforms. Social media is a place for scammer to fulfill their dreams because majority of people In day to day life are in social media like facebook, twitter, Instagram.etc., so it is easy for criminals to reach incredibly large numbers of people.

E-commerce scam

Fraudsters claim to be genuine online sellers, on sites such as facebook Marketplace. Consumers pay for goods, which then turn out to be counterfeit (e.g. fake clothing or gift vouchers) or poor quality (e.g. faulty or substandard). In some cases, goods simply never arrive.

Investment scam

Fraudsters advertise a ‘too good to be true’ investment opportunity, sometimes using news stories and advertisements that appear to be from genuine sources. Consumers who are tempted to invest lose some or all of their money.

Imposter scam-

Fraudsters pose as authentic brands, genuine friends or family, to gain a consumer’s trust asking them to purchase goods, send money or click on links which download malware to their computer.

E-commerce scams:

E-commerce on social media platforms is growing. And facebook Market-place, Instagram ads, and ‘buy buttons’ embedded in social media posts aim to create a seamless marketplace experience for consumers. Unfortunately, this creates further opportunities for scammers to exploit consumers, with our research showing that ecommerce14 scams are particularly prevalent on Facebook Marketplace and WhatsApp. E-commerce scams are particularly widespread in the emerging markets of Nigeria and India where fake vendors spread scams via social media forums and online marketplaces.

These typically involve the purchase of clothing items, with consumers receiving far-lower quality items than advertised or, in some cases, not receiving any item at all. Consumers mention these scams taking place on Facebook

Marketplace and WhatsApp. In higher income economies like the US and UK, e-commerce scams typically involve tech goods like mobile phones, or high cost items such as cameras or event tickets. Interviews with consumer protection organizations indicate that the most common type of e-commerce scam is sending items of much lower quality than described.

Different types of scams in e-commerce:

1.**Catfish:**

Frauds make fake profile and maintain a good relationships with people after gaining the trust they demand for money, bank-accounts or personal details.

2.**Cryptocurrency:**

Fake advertisements and fake messages which create the user to invest the money in bit coins.

3.**Clickbait Scam:**

Frauds try to create exciting link which tends the user to click on the links which leads to download malware in user’s computer.

4.**Membership Scams:**

Frauds invited to the fake groups in social media and ask victims to send money for membership.

5.**Quiz Scam:**

Scammers conduct a fake quiz and offer exciting gifts, to send the they will ask for address, phone number and soon .so, they can get some personal information from victims.

6.**Cash crabs:**

The consumer hacks the someone’s social media sites then ask money to their friends. Nowadays, this type of scam is very popular.

1.1 PROBLEM STATEMENT

As the social networking sites get more popular, spammers target these sites to spread spam posts the frauds are tending to give fake reviews and comments on website.

1.2 OBJECTIVE

The objective of this project is we propose a method using deep learning to detect spam's in social media.

1.3 SCOPE OF THE PROJECT

The main contributions of this project therefore are:

Data Analysis

Dataset Preprocessing

Training the Model

Testing of Dataset

CHAPTER 2

SYSTEM ANALYSIS AND DESIGN

2.1 EXISTINGSYSTEM

Twitter is the fastest growing social networking site among all social networks. An individual tweet is limited to 140 characters and tweets can be included only in text and HTTP connections. .Micro-Blogging services attracted not only legal users but also spammers. Spam is becoming a growing problem in online social networks like Twitter. Grier et al. have reported that 0.13% of the spam messages are posted on Twitter and is twice compared to the email spam As Twitter becomes an attractive platform for the spammers as the click rate on Twitter has increased day by day. In, this research the optimized text obtained from ABC algorithm are used to trained Naive Bayes and this approach is used to classify spam in the text.

2.2 PROPOSEDSYSTEM

It is often confusing for a person to decide which is spam or fake to detect fake messages or reviews in the massive collection of existing data. So, the system saves the time of a person by detecting the spam messages with the proposed algorithms.The main objective of the algorithm is to detect whether the messages are spam or not.There have been several algorithms implemented to detect the fake news or spam messages.

Here, we are having a dataset with reviews and ratings the reviews column contains all the comments related to the hotel whereas rating column consists 1 to 5 rating. One way of detecting spam is by removing “stopwords” and check the remaining words are relevant to the given circumstance. The project is about review of hotel consider words like: hotel, time, stay, service, etc. Most importantly spams will not contains appropriate words they are meaningless and irrelevant.

In this project we are using 4 algorithms

1.Logistic regression with 61% accuracy

2.Random Forest Algorithm with 52% accuracy

3.XGB Classifier Algorithm with 55% accuracy

4.LGBM Classifier with 60% accuracy.

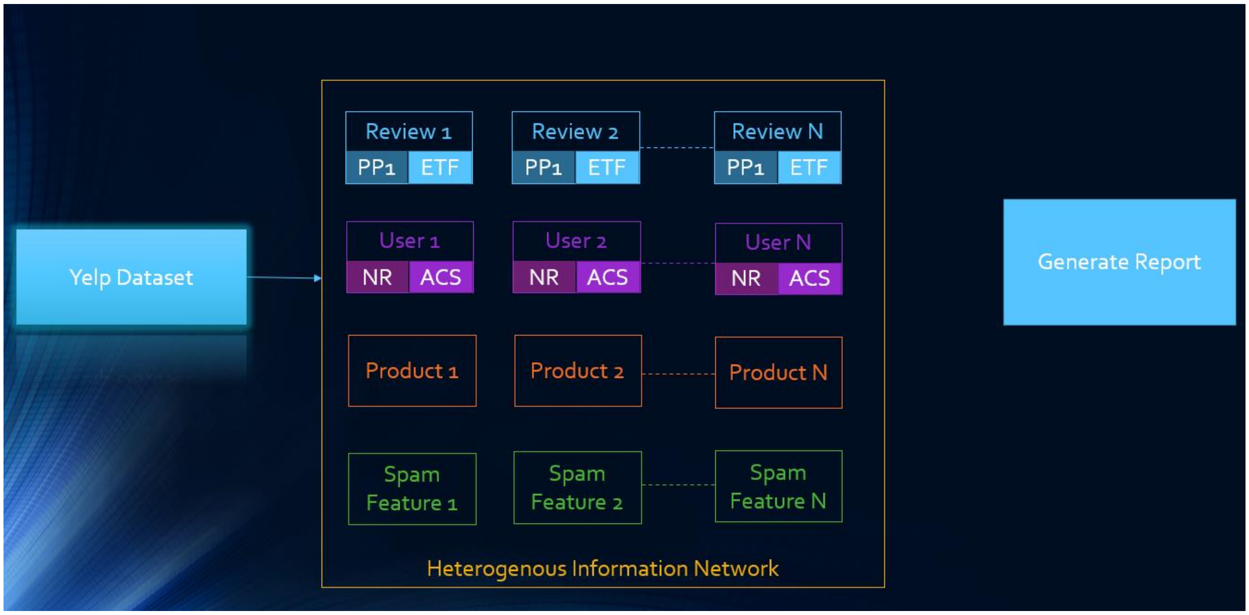


Fig2.2: Architectural diagram

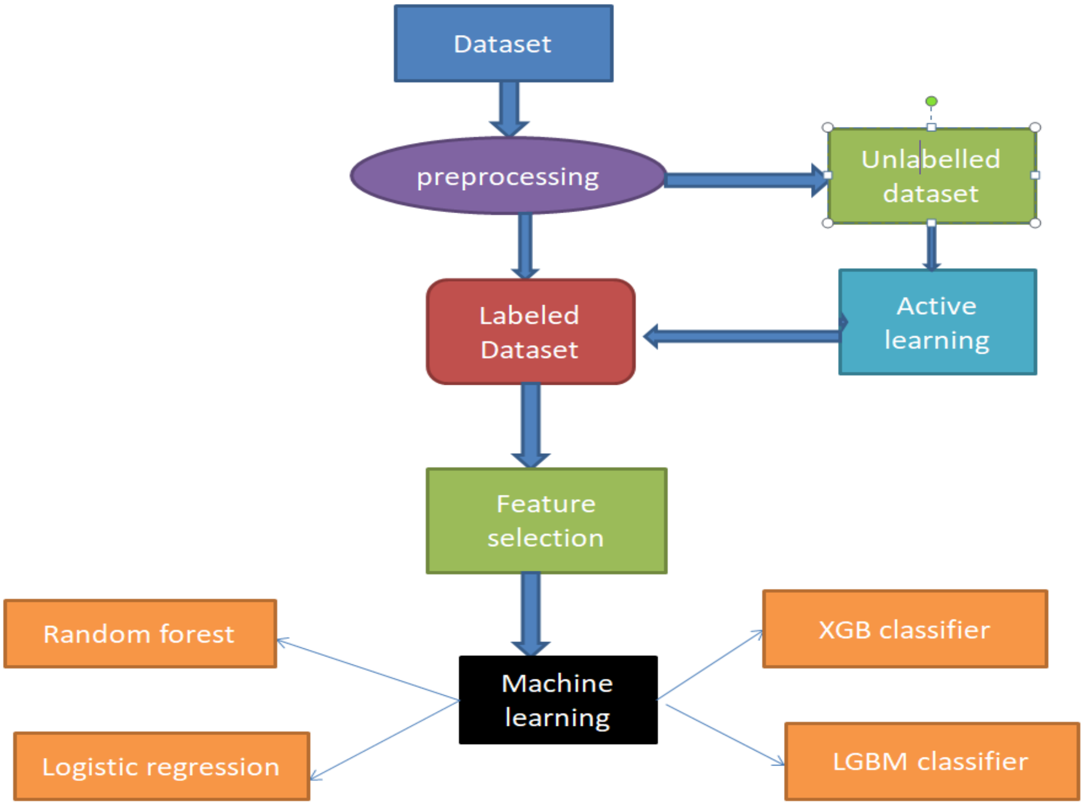


Fig2.2.2: Flow chart

CHAPTER 3
IMPLEMENTATION DETAILS

The libraries used in this project are:

3.1 Exploratory data analysis:

i. Pandas

Pandas is a popular Python package for data science, and with good reason: it offers powerful, expressive and flexible data structures that make data manipulation and analysis easy, among many other things. The DataFrame is one of these structures. Pandas is a high-level data manipulation tool developed by Wes McKinney. It is built on the Numpy package and its key data structure is called the DataFrame. DataFrames allow you to store and manipulate tabular

data in rows of observations and columns of variables.Pandas is built on top of the NumPy package, meaning a lot of the structure of NumPy is used or replicated in Pandas. Data in pandas is often used to feed statistical analysis in SciPy, plotting functions from Matplotlib, and machine learning algorithms in Scikit-learn. Jupyter Notebooks offer a good environment for using pandas to do data exploration and modeling, but pandas can also be used in text editors just as easily. Jupyter Notebooks give us the ability to execute code in a particular cell as opposed to running the entire file. This saves a lot of time when working with large datasets and complex transformations. Notebooks also provide an easy way to visualize pandas' DataFrames and plots. As a matter of fact, this article was created entirely in a Jupyter Notebook. There are two types of data structures in pandas: Series and DataFrames. Series: a pandas Series is a one-dimensional data structure ("a one dimensional ndarray") that can store values — and for every value it holds a unique index, too DataFrame: a pandas DataFrame is a two (or more) dimensional data structure – basically a table with rows and columns. The columns have names and the rows have indexes. Those who are familiar with R know the data frame as a way to store data in rectangular grids that can easily be overviewed. Each row of these grids corresponds to measurements or values of an instance, while each column is a vector containing data for a specific variable. This means that a data frame's rows do not need to contain, but can contain, the same type of values: they can be numeric, character, logical, etc.. Now, DataFrames in Python are very similar: they come with the Pandas library, and they are defined as two-dimensional labeled data structures with columns of potentially different types. In general, you could say that the Pandas DataFrame consists of three main components: the data, the index, and the columns. Firstly, the DataFrame can contain data that is: a Pandas

DataFrame a Pandas Series: a one-dimensional labeled array capable of holding any data type with axis labels or index. An example of a Series object is one column from a DataFrame a NumPy ndarray, which can be a record or structured a two-dimensional ndarray dictionaries of one-dimensional ndarray's, lists, dictionaries or Series. Some of the key features of Python Pandas are as follows: It provides DataFrame objects with default and customized indexing which is very fast and efficient. There are tools available for loading data of different fileformats into in-memory data objects. It is easy to perform data alignment and integrated handling of missing data in Python Pandas. It is very simple to perform pivoting and reshaping of data sets in Pandas. It also provides indexing, label-based slicing, and sub-setting of large data sets. We can easily insert and delete columns from a data structure. Data aggregation and transformations can be done using group by. High-performance merging and joining of data can be done using Pandas.It also provides time series functionality. Inserting and deleting columns in data structures. Merging and joining data sets. Reshaping and pivoting data sets. Aligning data and dealing with missing data. Manipulating data using integrated indexing for DataFrame objects. Performing split-apply-combine on data sets using the group by engine. Manipulating high-dimensional data in a data structure with a lower dimension using hierarchical axis indexing. Subsetting, fancy indexing, and label-based slicing data sets that are large in size. Generating data range, converting frequency, date shifting, lagging, and other time-series functionality. Reading from files with CSV, XLSX, TXT, among other formats. Arranging data in an order ascending or descending. Filtering data around a condition. Analyzing time series.
Iterating over a data set.

ii. Numpy

Numpy is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays. If you are already familiar with MATLAB, you might find this tutorial useful to get started with Numpy. A numpy array is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the rank of the array; the shape of an array is a tuple of integers giving the size of the array along each dimension. NumPy is, just like SciPy, Scikit-Learn, Pandas, etc. one of the packages that you just can't miss when you're learning data science, mainly because this library provides you with an array data structure that holds some benefits over Python lists, such as: being more compact, faster access in reading and writing items, being more convenient and more efficient. NumPy is a Python library that is the core library for scientific computing in Python. It contains a collection of tools and techniques that can be used to solve on a computer mathematical models of problems in Science and Engineering. One of these tools is a high-performance multidimensional array object that is a powerful data structure for efficient computation of arrays and matrices. To work with these arrays, there’s a vast amount of high-level mathematical functions operate on these matrices and arrays.an array is basically nothing but pointers. It’s a combination of a memory address, a data type, a shape, and strides: The data pointer indicates the memory address of the first byte in the array, The data type or dtype pointer describes the kind of elements that are contained within the array, The shape indicates the shape of the array, and The strides are the number of bytes that should be skipped in memory to go to the next element. If your strides are (10,1), you need to proceed one byte to get to the next column and 10 bytes to locate the next rowor in other words, an array contains information about the raw data, how to locate an element and how to interpret an element. With NumPy, we work with multidimensional arrays. We’ll dive into all of the possible types of multidimensional arrays later on, but for now, we’ll focus on 2-dimensional arrays. A 2-dimensional array is also known as a matrix, and is something you should be familiar with. In fact, it’s just a different way of thinking about a list of lists. A matrix has rows and columns. By specifying a row number and a column number, we’re able to extract an element from a matrix. We can create a NumPy array using the numpy array function. If we pass in a list of lists, it will automatically create a NumPy array with the same number of rows and columns. Because we want all of the elements in the array to be float elements for easy computation, we’ll leave off the header row, which contains strings. One of the limitations of NumPy is that all the elements in an array have to be of the same type,so if we include the header row, all the elements in the array will be read in as strings. Because we want to be able to do computations like find the average quality of the wines, we need the elements to all be floats. NumPy has several advantages over using core Python mathematical functions, a few of which are outlined here: NumPy is extremely fast when compared to core Python thanks to its heavy use of C extensions. Many advanced Python libraries, such as Scikit-Learn, Scipy, and Keras, make extensive use of the NumPy library. Therefore, if you plan to pursue a career in data science or machine learning, NumPy is a very good tool to master. NumPy comes with a variety of built-in functionalities, which in core Python would take a fair bit of custom code.

iii. Matplotlib

Plotting of data can be extensively made possible in an interactive way by Matplotlib, which is a plotting library that can be demonstrated in Python scripts. Plotting of graphs is a part of data visualization, and this property can be achieved by making use of Matplotlib.

Matplotlib makes use of many general-purpose GUI toolkits, such as wxPython, Tkinter, QT, etc., in order to provide object-oriented APIs for embedding plots into applications. John D. Hunter was the person who originally wrote Matplotlib, and its lead developer was Michael Droettboom. One of the free and open-source Python library which is basically used for technical and scientific computing is Python SciPy. Matplotlib is widely used in SciPy as most scientific calculations require plotting of graphs and diagrams. Matplotlib is a plotting library like GNU plot. The main advantage towards GNU plot is the fact that Matplotlib is a Python module. Due to the growing interest in python the popularity of matplotlib is continually rising as well.

Another reason for the attractiveness of Matplotlib lies in the fact that it is widely considered to be a perfect alternative to MATLAB, if it is used in combination with Numpy and Scipy. Whereas MATLAB is expensive and closed source, Matplotlib is free and open source code. It is also object-oriented and can be used in an object oriented way. Further more it can be used with general-purpose GUI tool kits like wx Python, Qt, and GTK+. There is also a procedural "pylab", which designed to closely resemble that of MATLAB. This can make it extremely easy for MATLAB users to migrate to matplotlib. Matplotlib can be used to create publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

Another characteristic of matplotlib is its steep learning curve, which means that users usually make rapid progress after having started. The official website has to say the following about this: "matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatterplots, etc, with just a few lines of code."

3.2 Deep Learning

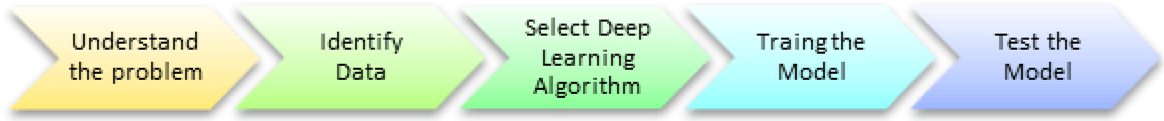
Deep learning is a computer software that mimics the network of neurons in a brain. It is a subset of machine learning and is called deep learning because it makes use of deep neural networks.

Deep learning algorithms are constructed with connected layers.

1.The first layer is called the Input Layer

2.The last layer is called the Output Layer

All layers in between are called Hidden Layers. The word deep means the network join neurons in more than two layers. A deep neural network provides state-of-the-art accuracy in many tasks, from object detection to speech recognition. They can learn automatically, without predefined knowledge explicitly coded by the programmers.



To grasp the idea of deep learning, imagine a family, with an infant and parents. The toddler points objects with his little finger and always says the word 'cat.' As its parents are concerned about his education, they keep telling him 'Yes, that is a cat' or 'No, that is not a cat.' The infant persists in pointing objects but becomes more accurate with 'cats.' The little kid, deep down, does not know why he can say it is a cat or not. He has just learned how to hierarchies complex features coming up with a cat by looking at the pet overall and continue to focus on details such as the tails or the nose before to make up his mind. A neural network works quite the same. Each layer represents a deeper level of knowledge, i.e., the hierarchy of knowledge. A neural network with four layers will learn more complex feature than with that with two layers.

The learning occurs in two phases.

1.The first phase consists of applying a nonlinear transformation of the input and create a statistical model as output.

2.The second phase aims at improving the model with a mathematical method known as derivative. The neural network repeats these two phases hundreds to thousands of time until it has reached a tolerable level of accuracy. The repeat of this two-phase is called an iteration

Classification of Neural Networks

1.Shallow neural network: The Shallow neural network has only one hidden layer between the input and output.

2.Deep neural network: Deep neural networks have more than one layer. For instance, Google LeNet model for image recognition counts 22 layers. The computational models in Deep Learning are loosely inspired by the human brain. The multiple layers of training are called Artificial Neural Networks (ANN).

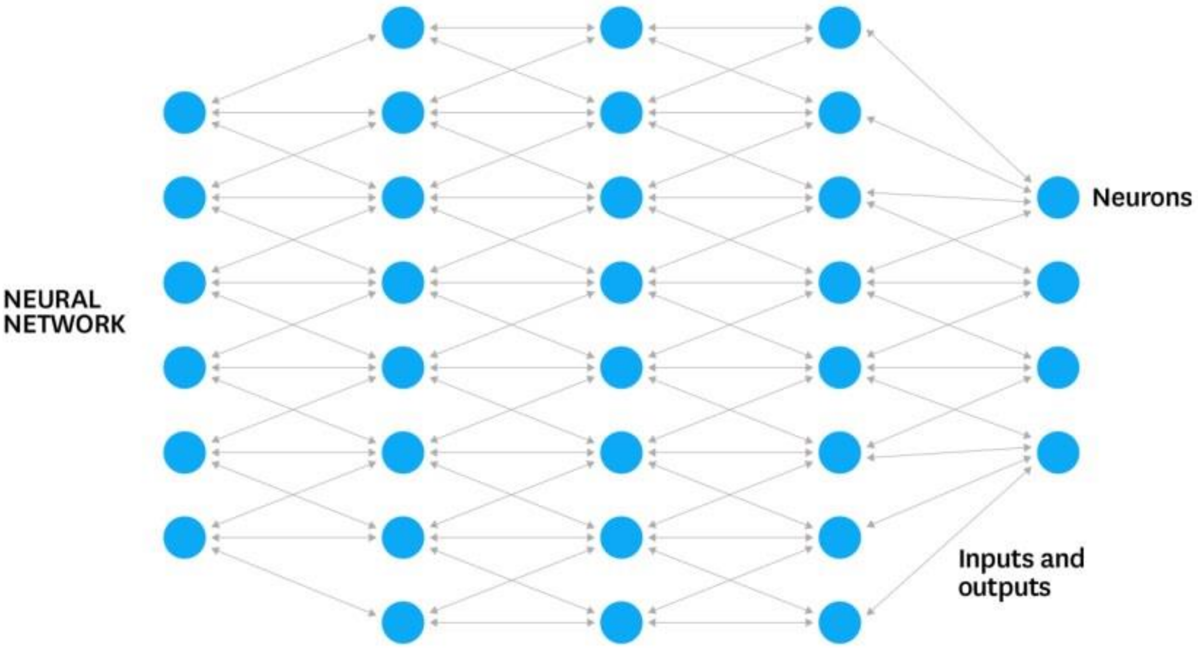


Fig3.2: Neural network

Neuron

Artificial Neural Networks contain layers of neurons. A neuron is a computational unit that calculates a piece of information based on weighted input parameters. Inputs accepted by the neuron are separately weighted. Inputs are summed and passed through a non-linear function to produce output. Each layer of neurons detects some additional information, such as edges of things in a picture or tumors in a human body. Multiple layers of neurons can be used to detect additional information about input parameters.

Nodes

Artificial Neural Network is an interconnected group of nodes akin to the vast network of layers of neurons in a brain. Each circular node represents an artificial neuron and an arrow represents a connection from the output of one neuron to the input of another.

Inputs

Inputs are passed into the first layer. Individual neurons receive the inputs, with each of them receiving a specific value. After this, an output is produced based on these values.

Outputs

The outputs from the first layer are then passed into the second layer to be processed. This continues until the final output is produced. The assumption is that the correct output is predefined. Each time data is passed through the network, the end result is compared with the correct one, and tweaks are made to their values until the network creates the correct final output each time.

Some of the commonly used neural networks are as follows:

Artificial Neural Network (ANN)

Convolutional Neural Network (CNN)

Recurrent Neural Network (RNN)

Deep Neural Network (DNN)

Deep Belief Network (DBN)

Artificial neural networks are one of the main tools used in machine learning. As the "neural" part of their name suggests, they are brain-inspired systems which are intended to replicate the way that we humans learn. Neural networks consist of input and output layers, as well as (in most cases) a hidden layer consisting of units that transform the input into something that the output layer can use. They are excellent tools for finding patterns which are far too complex or numerous for a human programmer to extract and teach the machine to recognize.

Generally, the working of a human brain by making the right connections is the idea behind ANNs. That was limited to use of silicon and wires as living neurons and dendrites.

Here, neurons, part of human brain. That was composed of 86 billion nerve cells. Also, connected to other thousands of cells by Axons. Although, there are various inputs from sensory organs. That was accepted by dendrites. As a result, it creates electric impulses. That is used to travel through the Artificial neural network. Thus, to handle the different issues, neuron send a message to another neuron. As a result, we can say that ANNs are composed of multiple nodes. That imitate biological neurons of the human brain. Although, we connect these neurons by links. Also, they interact with each other. Although, nodes are used to take input data. Further, perform simple operations on the data. As a result, these operations are passed to other neurons. Also, output at each node is called its activation or node value.

3.3 Algorithms:

i. Logistic regression:

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist.

In regression analysis, logistic regression (or logistic regression) is estimating the parameters of a logistic model (a form of binary regression).

It is a supervised learning technique. It predicts categorical data, the output is form of yes or no,0 or 1.

Logistic regression is similar to linear regression. Linear regression is solve regression problems, whereas logistic regression to solve classification problem. In logistic regression ‘s’ shape logistic function is used to predict max value which means output generation 0 or 1. logistic regression indicates likelihood or not based on this we draw a logistic curve. It mainly for observing the data from dataset and convert into different types of classifications. so, that we can know most effect variable and least effect variable.

Logistic Function (Sigmoid Function):

1.The sigmoid function is a mathematical function used to map the predicted values to probabilities’

2.It maps any real value into another value with range of 0 and 1.

3.The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the “s” form. The S-form curve is called the sigmoid or logistic function.

4.In logistic regression ,we use the concept of threshold value, which defines the probabilities of either 0 or 1.Such as values above the threshold value tends to 1,and a value below the threshold value tends to 1.

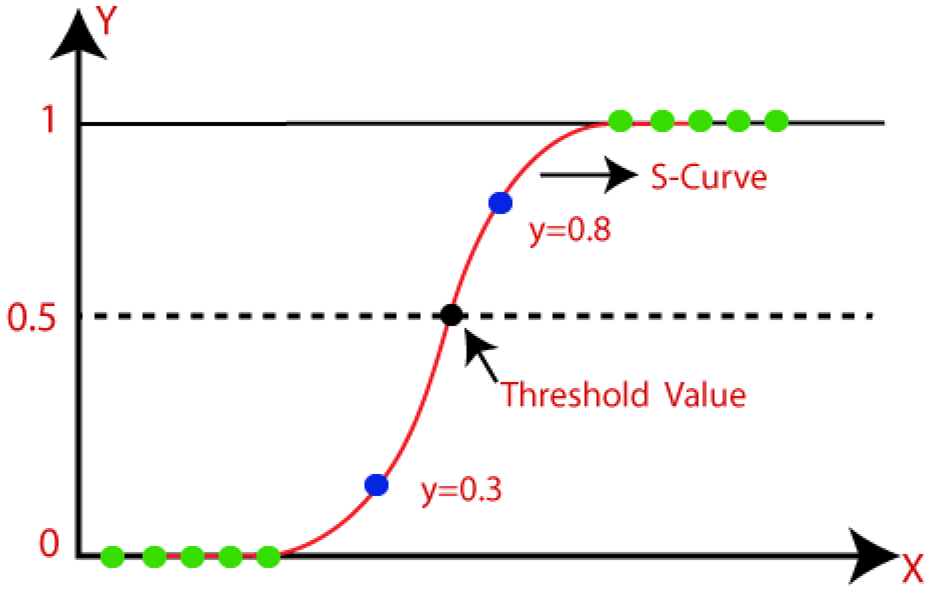


Fig3.3.1: Logistic Regression

Assumptions for logistic regression:

1.The dependent variable must be categorical in nature.

2.The independent variable should not have multi-collinearity.

3.We know the equation of straight line can be written as

Y=b0+b1x1+b2x2+b3x3+………+bnxn

4.In logistic regression y can be between 0 and 1 only, so for this let’s divide the above equation by(1-y)

y/(1-y); 0 for y=0, and infinity for y=1

5.But we need range between –infinity to +infinity the take logarithm of the equation it will become

Log(y/(1-y))=b0++b2x2+b3x3+………+bnxn

Types of logistic regression:

1.Binary logistic regression

2.Multinomial logistic regression

3.Ordinal logistic regression

Steps:

1.Data pre-processing step

Import libraries and dataset, extracting the independent and dependent variables, then split the dataset.

2.Fitting logistic regression to the Training set

Import logistic regression to from sklearn library then fit the logistic regression to the training set.

3.predicting the test results:

Y_pred =classifier.predict(x_test)

4.Test the accuracy of the result:

First import confusion matrix from sklearn matrix

Import sklearn.metrics import confusion_matrix

Cm=confusion_matrix ----to check accuracy

Random Forest classification:

Random Forest classification is a popular machine learning algorithm .it used supervised technique.it can solve classification and regression problems. Mainly it is used for classification. By using a dataset we can derive a Decision tree. The combination of many Decision trees is known as Random Forest classification. Each decision tree has individual output based on majority voting it can predict the final class as shown in the figure. If a dataset it is having more Decision trees then we can predict more accuracy and there is no overfitting.

Assumption for random forest

1.There should be some actual values in the features variables of the dataset so that the classifier

2.can predict accurate results rather than a guessed result.

3.The predictions from each tree must have very low correlation.

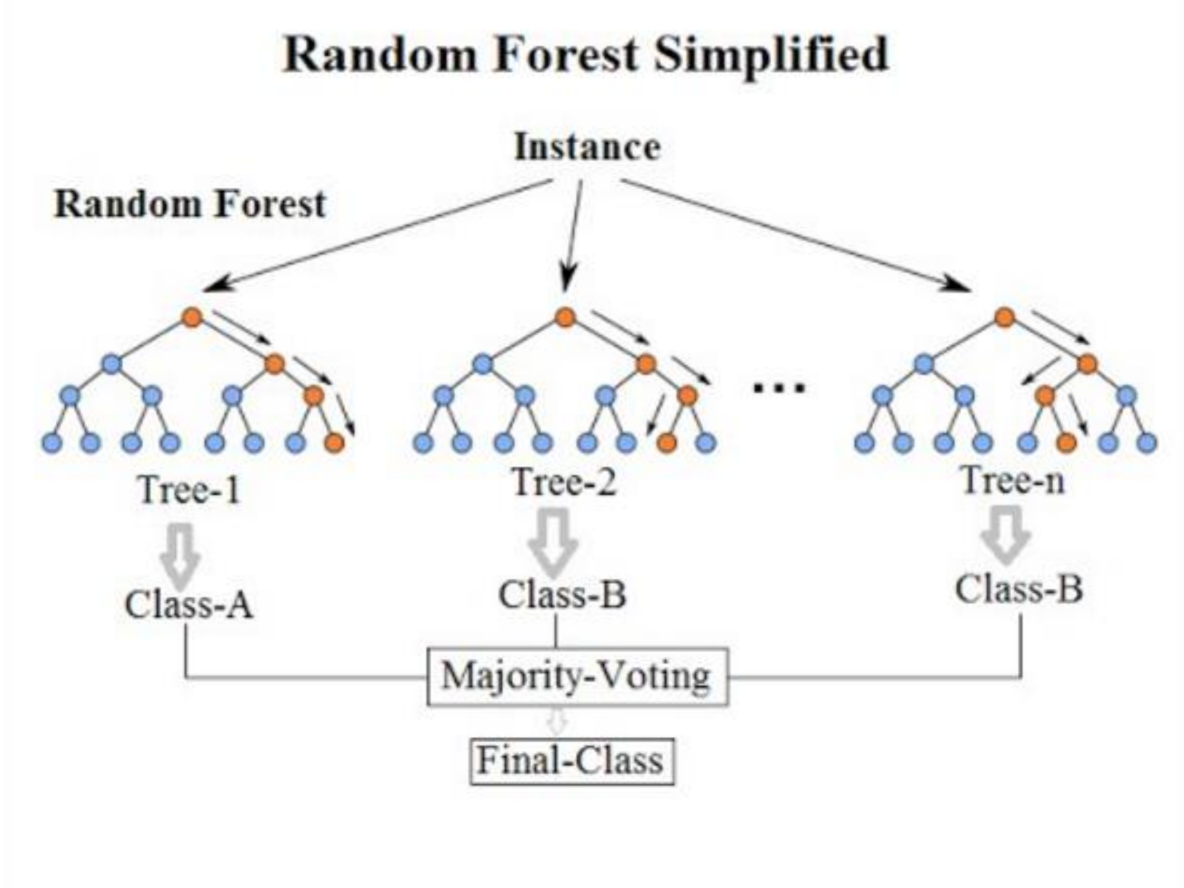


Fig3.3.2:Random Forest

Why use random forest?

1.It takes less training times as compared to other algorithms.

2.It predicts output with high accuracy,even for the large dataset it runs efficiently.

3.It can also maintain accuracy when a large proportion of data is missing.

How does random forest algorithm work?

Step1: Select random k data points from the training set.

Step 2: Build the decision trees associated with the selected data points(subsets)

Step 3: Choose the number N for decision trees that you want to built

Step 4: Repeat step 1&2

Step 5: For new data points , find the predictions of each decision tree, and assign the new data points to the category that win the majority votes.

Application of random forest

1.Banking: Banking sector mostly uses this algorithm for identification of loan risks

2.Medicine: With the help of this algorithm, diseases trends and risks of the disease can be identified.

3.Land use: We can identify the areas of similar land use by this algorithm

4.Marketing: Marketing trends can be identified using this algorithm.

Demerits:

Although random forest can be used for both classification and regression tasks , it is not more suitable for Regression task.

3.XGBCLASSIFIER:

It is also known as Xtream Gradient Boosting or XGBoost. It is a supervised learning.

There are 2 Ensemble techniques

1.Bagging or Bootstrapping aggregation

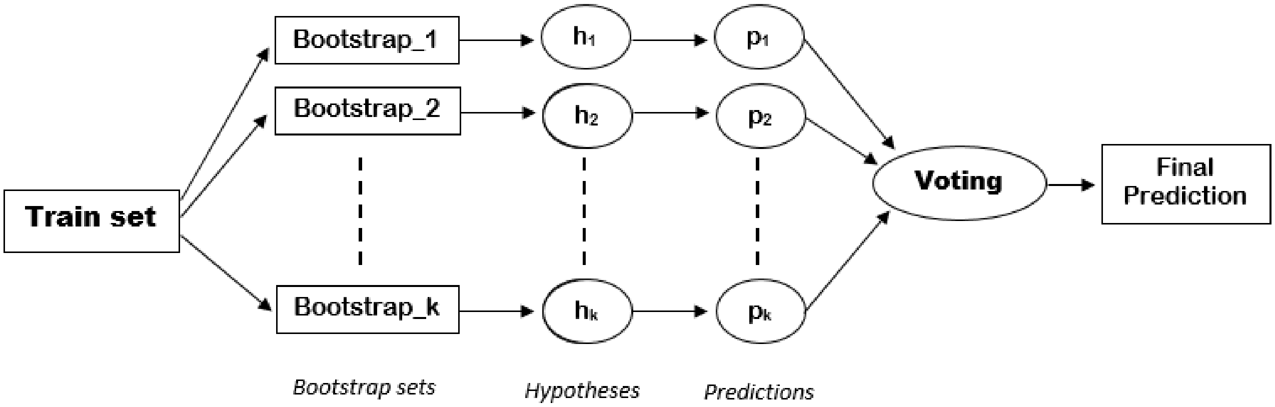


Fig 3.3.3: Bagging or Bootstrapping aggregation

2.Boosting technique

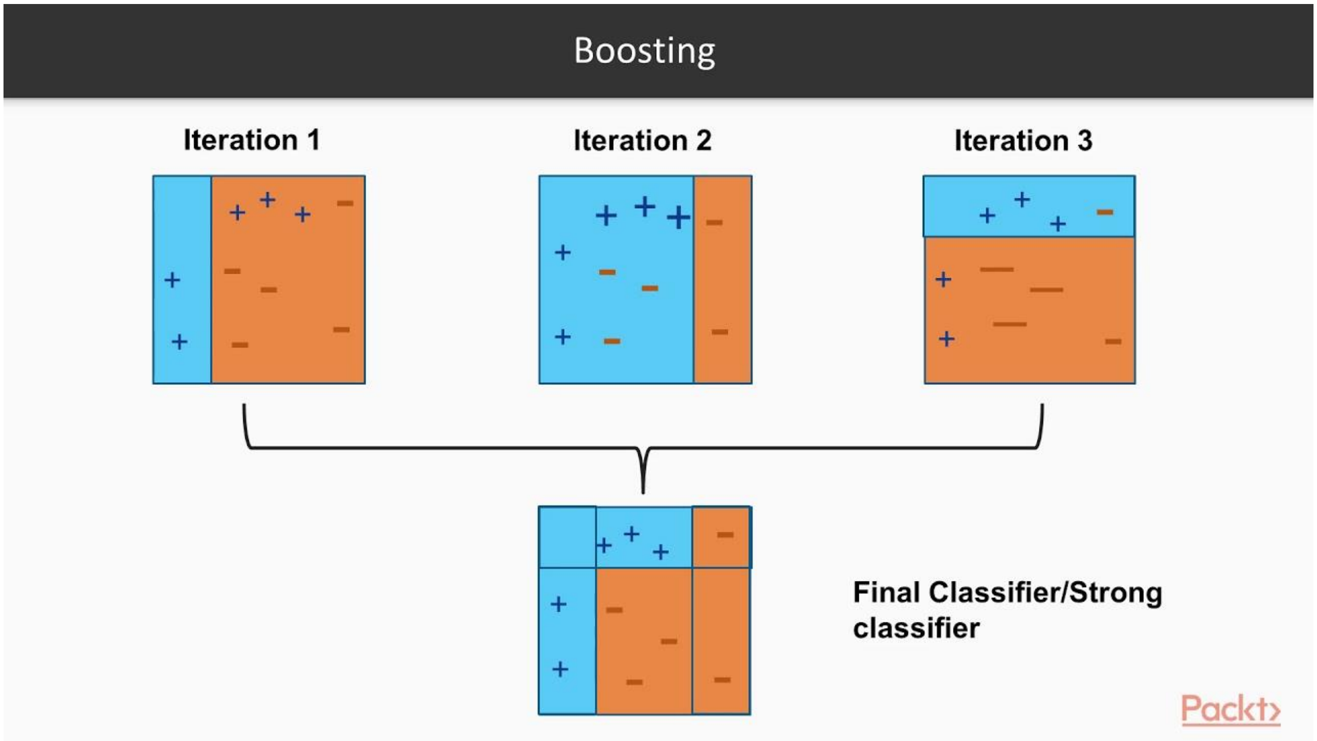


Fig3.3.3:Boosting Technique

4.LGBM CLASSIFIER ALGORITHM:

It is also known as LightGBM classifier. LightGBM is a gradient boosting framework that uses tree-based learning algorithms. It is designed to be distributed and efficient with the following **Advantages:**

1.Faster training speed and higher efficiency.

2.Lower memory usage.

3.Better accuracy.

4.Support of parallel, distributed, and GPU learning.

5.Capable of handling large-scale data.

*6.it*splits the tree leaf wise**.**

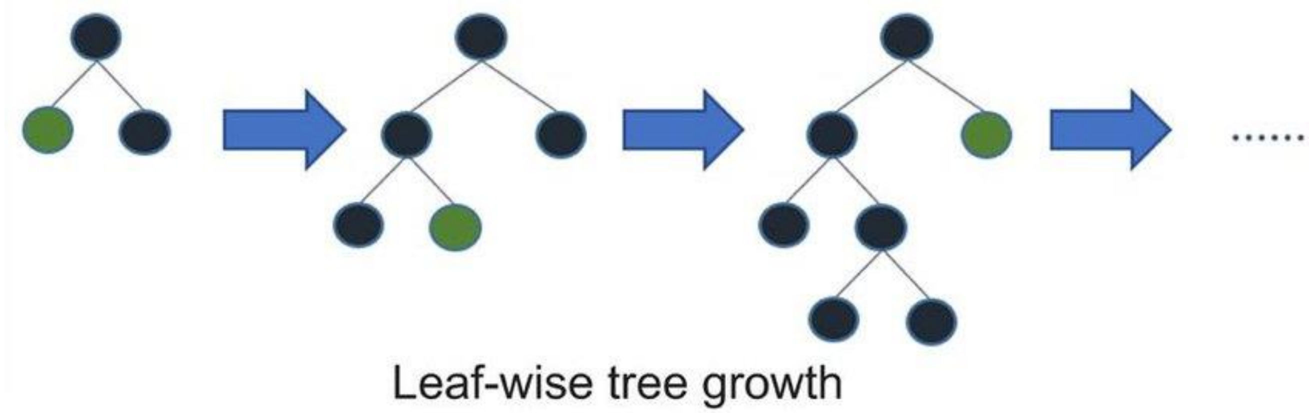


Fig3.3.4:leaf-wise tree growth

Leaf-wise may cause over-fitting when is small, so LightGBM includes the parameter to limit tree depth. However, trees still grow leaf-wise even when is specified.Reduced cost of calculating the gain for each split. Pre-sort-based algorithms have time complexity. Computing the histogram has time complexity, but this involves only a fast sum-up operation. Once the histogram is constructed, a histogram-based algorithm has time complexity, and is far smaller than. Use histogram subtraction for further speed up to get one leaf's histograms in a binary tree, use the histogram subtraction of its parent and its neighbor. So it needs to construct histograms for only one leaf (with smaller than its neighbor). It then can get histograms of its neighbor by histogram subtraction with small cost Reduce memory usage. Replaces continuous values with discrete bins. If is small, can use small data type, e.g: uint8_t, to store training data. No need to store additional information for pre-sorting feature values.

3.4 List of modules:

Module 1: Dataset Preprocessing

In any Machine Learning process, Data Preprocessing is that step in which the data gets transformed, or Encoded, to bring it to such a state that now the machine can easily parse it. In other words, the features of the data can now be easily interpreted by the algorithm.A dataset can be viewed as a collection of data objects, which are often also called as a records, points, vectors, patterns, events, cases, samples, observations, or entities. Data objects are described by a number of features, that capture the basic characteristics of an object, such as the mass of a physical object or the time at which an event occurred, etc.. Features are often called as variables, characteristics, fields, attributes, or dimensions. The first pre-processing technique is remove @ which means it scan the whole document of input dataset and after comparing it with @ it deletes @ from every available comment with @.The next step of pre-processing is remove URL where the whole input document gets scanned and compared with http:\\... and the comments having URL are deleted. Further we move on to stop word removal being the step in data pre-processing. Stopword removal exactly means that from the whole statement after scanning it removes the words like and, is, the, etc and only keeps noun and adjective. Tokenization and Normalization are carried out thereafter. Porter Stemmer Algorithm is used thereafter. The Porter stemming algorithm (or 'Porter stemmer') is a process for removing the commoner morphological and inflexional endings from words in English. Its main use is as part of a term normalization process that is usually done when setting up Information Retrieval systems.

Module 2 : Analysis of Emails

Emails are available only for a short time approximately seven days, after being posted .Many real time spam detection systems that rely on historical features from past emails, are affected by this constraint and may be practically less effective. Readily available, dynamic features offer an enhanced opportunity to distinguish spam from non-spam emails. Pairwise engagement features sub categorized into: Engage-with Features (EwF) include features that describe user activities on Email and users can influence or choose how to alter their values. Features under this group include friends count, statuses count, email type, email creation time, email creation frequency, etc. Engaged-by Features (EbF) are similar to features in the EwFgroup. The main difference is that features under this group cannot be influenced by users directly. For instance, a user relies on other users to increase their favorites count or to attract more followers. Features in this group include count, favorites count, number of forward emails, etc.

Module 3: Feature Selection

Categorical: Features whose values are taken from a defined set of values. For instance, days in a week: {Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday} is a Category because its value is always taken from this set. Another example could be the Boolean set : {True, False}Numerical: Features whose values are continuous or integer-valued. They are represented by numbers and possess most of the properties of numbers. For instance, number of steps you walk in a day, or the speed at which you are driving your car at. The Email platform facilitates global connections and interactions of diverse users. The relevant attributes that enable users to connect and form the basis of our feature extraction. Furthermore, features can be classified as basic features or derived features. The aforementioned features are basic features, whereas derived features are computed using two or more basic features or are based on further analysis, e.g: sentiment analysis or entropy computation on textual data. Features can also be characterized as static or dynamic. Static features cannot be changed once the accountis created e.g: user ID and account creation time, where as dynamic features keep changing depending on the user sleve of engagements on Email e.g. statuses count.

Module 4:SpamPrediction

Convolutional Neural Networks have numerous applications beyond image recognition. For example, Neural Network's have predictive power for time series forecasting and natural language processing (NLP). The input to a Neural Network is a matrix. In image recognition, each image's pixels are coded as numerical values representing the intensity of color for each pixel. We'll focus on the NLP application of CNN sand train a Word Neural Network. A Word Neural Network's input matrix includes rows representing words in a sentence and columns representing word embeddings of 'n' dimensions. Keras makes it easy to create a Word Neural Network in just a few lines of code. For this model, we generate embeddings within our corpus using the Keras "embedding" layer. Note that the output from the embedding layer is a matrix, which is the necessary input to the convolutional layer.

CHAPTER-4

CONCLUSION

In recent years, review spam detection has received significant attention both business and academia due to the potential impact the reviews can have on consumer behavior and purchasing decisions. Supervised learning is the most frequent machine learning approach for performing review spam detection; however, obtaining labeled reviews for training is difficult and manual identification of fake reviews has poor accuracy. The accurate detection of spam is a big issue, and many detection methods have been proposed by various researchers. Thus, the results suggest that the proposed method is more reliable for accurate and on-time detection of spam, and it will secure the communication systems of messages and e-mails

7.REFERENCES

[1] B. L. Adokshaja ,S. J. Saritha“ Third party public auditing on cloud storage using the cryptographic algorithm” published on: 2017.

[2] L Krithikashree; S. Manisha ; M Sujithra "Audit cloud: ensuring data integrity for mobile devices in cloud storage" published on:2018.

[3]Yanqun Zhang "A flexible distributed storage integrity auditing mechanism in Cloud Computing"

Published on:2009

[4] Filipe Apolinário ; Miguel Pardal ; Miguel Correia "S-Audit: Efficient Data Integrity Verification for Cloud Storage" published on:2018

[5]**https://ieeexplore.ieee.org/document/8275931**

[6]https://www.researchgate.net/publication/318861757_Spam_E-Mail_Classification_by_Utilizing_N-Gram_Features_of_Hyperlink_Texts

[7] Spam E-Mail Classification by Utilizing N-Gram Features of Hyperlink TextDetecting Algorithmically Generated Malicious Domain Names http://eprints.networks.imdea.org/67/1/Detecting_Algorithmically_Generated_Malicious_Domain_Names_-_2010_EN.pdf

[8] Spam E-Mail Classification by Utilizing N-Gram Features of Hyperlink Texts**https://pdfs.semanticscholar.org/c5d5/9b58c2d46e692429446eec10407911b0a416.pdf**

[9] Time-efficient spam e-mail filtering using n-gram models.

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.136.2510&rep=rep1&type=pdf

[10] WORDS VS.CHARACTER N-GRAMS FOR ANTI-SPAM FILTERING.

http://www.icsd.aegean.gr/Stamatatos/papers/IJAIT-spam.pdf

8.APPENDIX-1

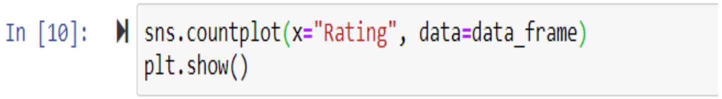
```
import sklearn
import re
import string
import warnings
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from nltk.tokenize import word_tokenize
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.svm import SVC
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
warnings.filterwarnings("ignore")
pip install lightgbm
pip install --user imblearn
pip install xgboost
data_frame = pd.read_csv("tripadvisor_hotel_reviews.csv")
data_frame.head()
data_frame.info()
```

```
data_frame["Rating"].value_counts()
sns.countplot(x="Rating", data=data_frame)
plt.show()
data_frame = data_frame.sample(frac=1).reset_index(drop=True)
# Remove special characters from the sentence
def clean_text(sentence):
# Convert to lower case
sentence = sentence.lower()
# split the sentence
sentence = sentence.split()
# Join the sentence
sentence = " ".join(sentence)
# Remove special characters from the sentence
sentence = re.sub(f'[{re.escape(string.punctuation)}]', "", sentence)
return sentence
data_frame["Review"] = data_frame["Review"].apply(clean_text)
x_train, x_test, y_train, y_test = train_test_split(data_frame["Review"], data_frame["Rating"],test_size=0.2, random_state=42)
# Apply Tfidf Vectorizer to convert sentence to tokens
tfidf = TfidfVectorizer(tokenizer=word_tokenize, token_pattern=None)
tfidf.fit(data_frame["Review"])
x_train_vector = tfidf.transform(x_train)
x_test_vector = tfidf.transform(x_test)
# Classes are imbalanced
# SMOTE to over sample and balance classes.
x_smote, y_smote = SMOTE().fit_resample(x_train_vector, y_train)
import nltk
```

```
nltk.download('punkt')
def evaluation_metric(y_test, y_hat, model_name):
accuracy = accuracy_score(y_hat, y_test)
print("Model: ", model_name)
print("\nAccuracy: ", accuracy)
print(classification_report(y_hat, y_test))
plt.figure(figsize=(10,6))
sns.heatmap(confusion_matrix(y_hat, y_test), annot=True, fmt=".2f")
plt.show()
 return accuracy
lr_model = LogisticRegression()
lr_model.fit(x_smote, y_smote)
lr_preds = lr_model.predict(x_test_vector)
lr_accuracy = evaluation_metric(lr_preds, y_test, "Logistic Regression")
rf_model = RandomForestClassifier()
rf_model.fit(x_smote, y_smote)
rf_preds = rf_model.predict(x_test_vector)
rf_accuracy = evaluation_metric(rf_preds, y_test, "Random Forest Classifier")
xgb_model = XGBClassifier(max_depth=10,random_state=1,learning_rate=0.05,seed=1)
xgb_model.fit(x_smote, y_smote)
xgb_preds = xgb_model.predict(x_test_vector)
xgb_accuracy = evaluation_metric(xgb_preds, y_test, "XGB Classifier")
lgb_model = LGBMClassifier()
lgb_model.fit(x_smote, y_smote)
lgb_preds = lgb_model.predict(x_test_vector)
```

```
lgb_accuracy = evaluation_metric(lgb_preds, y_test, "LGBM Classifier")
x = ["Random Forest", "Logistic Regression", "XGB Classifier", "LGBM Classifier"]
y = [rf_accuracy, lr_accuracy, xgb_accuracy, lgb_accuracy]
plt.bar(x=x, height=y)
plt.title("Algorithm Accuracy Comparison")
plt.xticks(rotation=15)
plt.xlabel("Algorithms")
plt.ylabel("Accuracy")
plt.show()
```

9.APPENDIX-2



```
In [10]:  sns.countplot(x="Rating", data=data_frame)
          plt.show()
```

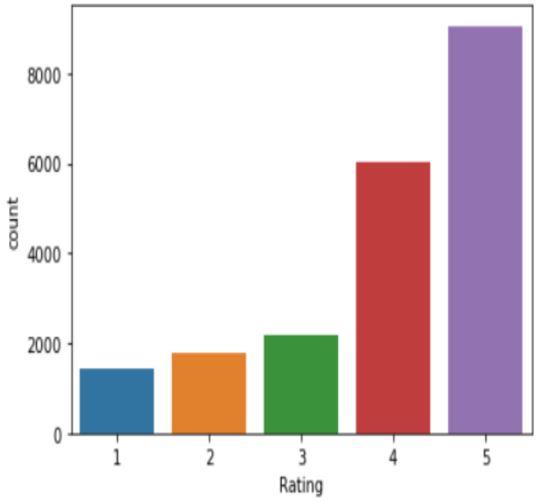
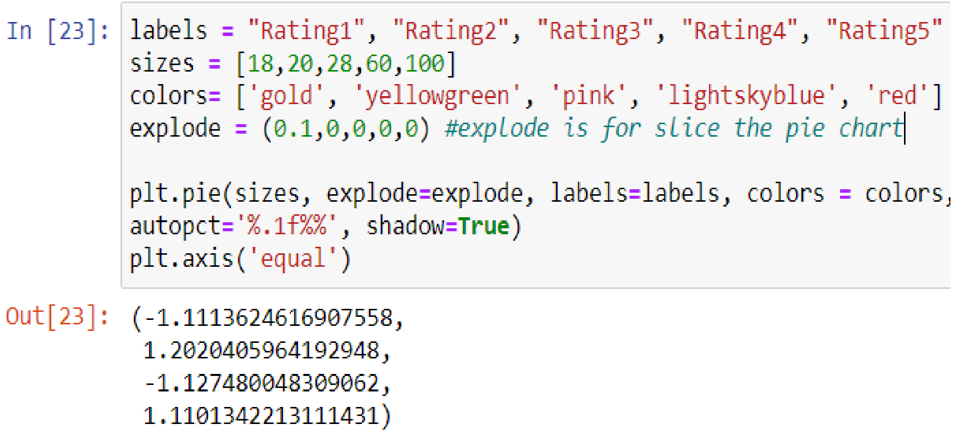


Fig 9.1:bar-chart



```
In [23]:  labels = "Rating1", "Rating2", "Rating3", "Rating4", "Rating5"
          sizes = [18,20,28,60,100]
          colors= ['gold', 'yellowgreen', 'pink', 'lightskyblue', 'red']
          explode = (0.1,0,0,0,0) #explode is for slice the pie chart

          plt.pie(sizes, explode=explode, labels=labels, colors = colors,
          autopct='%.1f%%', shadow=True)
          plt.axis('equal')

Out[23]:  (-1.1113624616907558,
           1.2020405964192948,
           -1.127480048309062,
           1.1101342213111431)
```

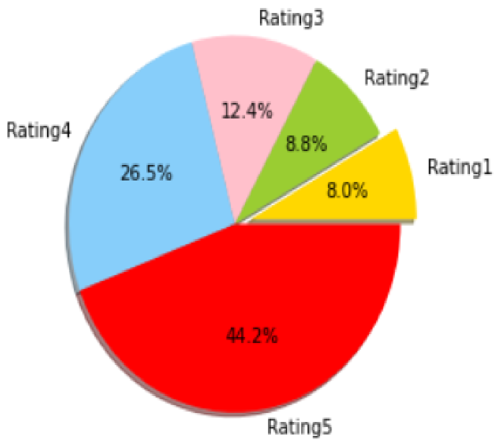
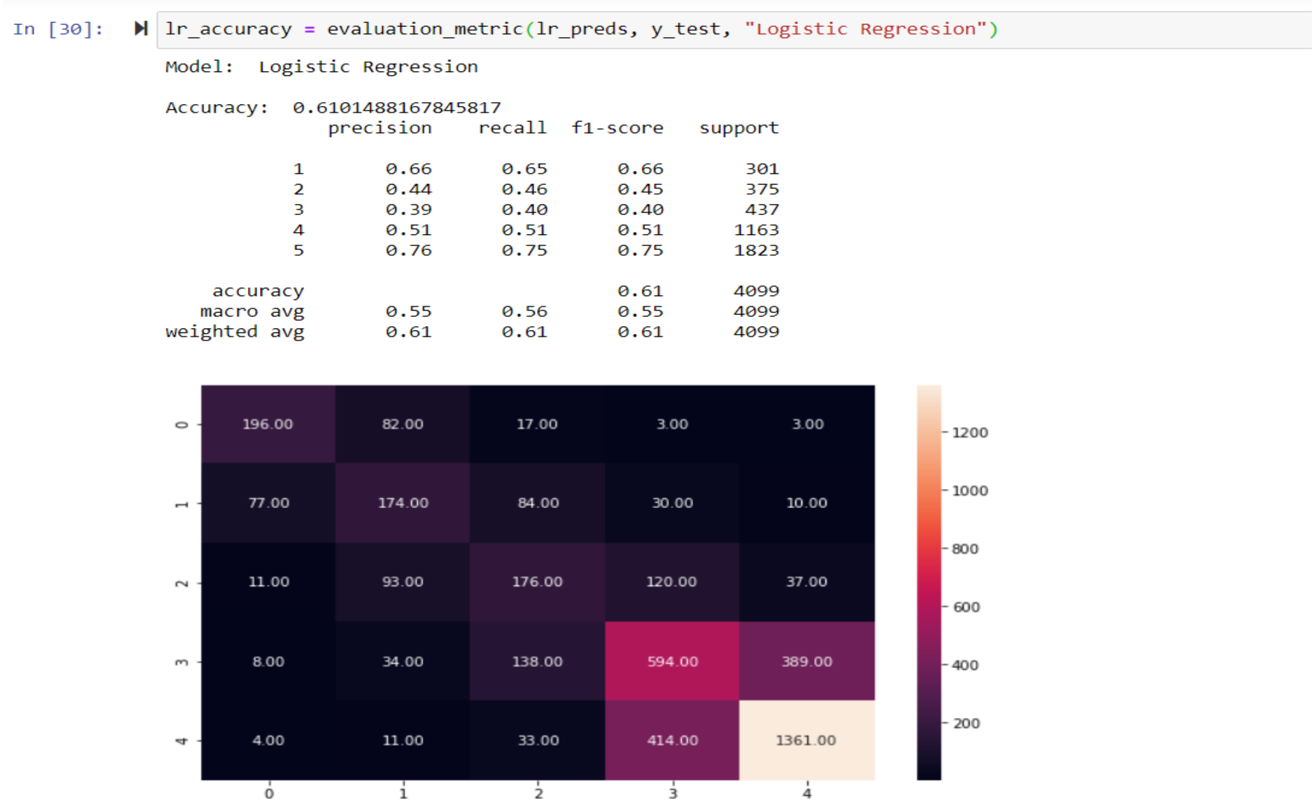


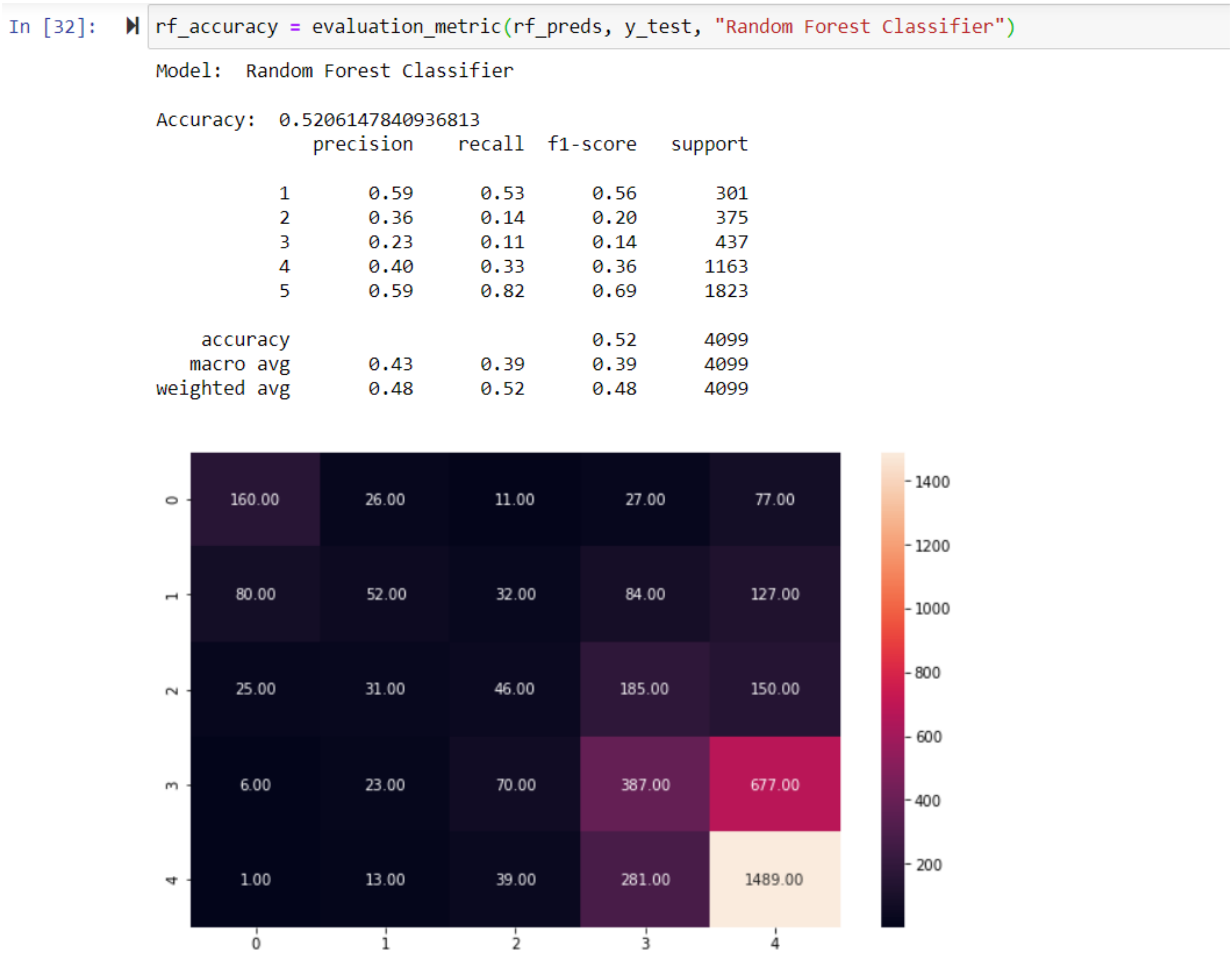
Fig 9.2:pie-chart



```
In [30]:  lr_accuracy = evaluation_metric(lr_preds, y_test, "Logistic Regression")
```

```
Model:  Logistic Regression

Accuracy:  0.6101488167845817
              precision    recall  f1-score   support

           1       0.66      0.65      0.66       301
           2       0.44      0.46      0.45       375
           3       0.39      0.40      0.40       437
           4       0.51      0.51      0.51      1163
           5       0.76      0.75      0.75      1823

    accuracy                           0.61      4099
   macro avg       0.55      0.56      0.55      4099
weighted avg       0.61      0.61      0.61      4099
```

Fig 9.3 logistic regression accuracy and heat map

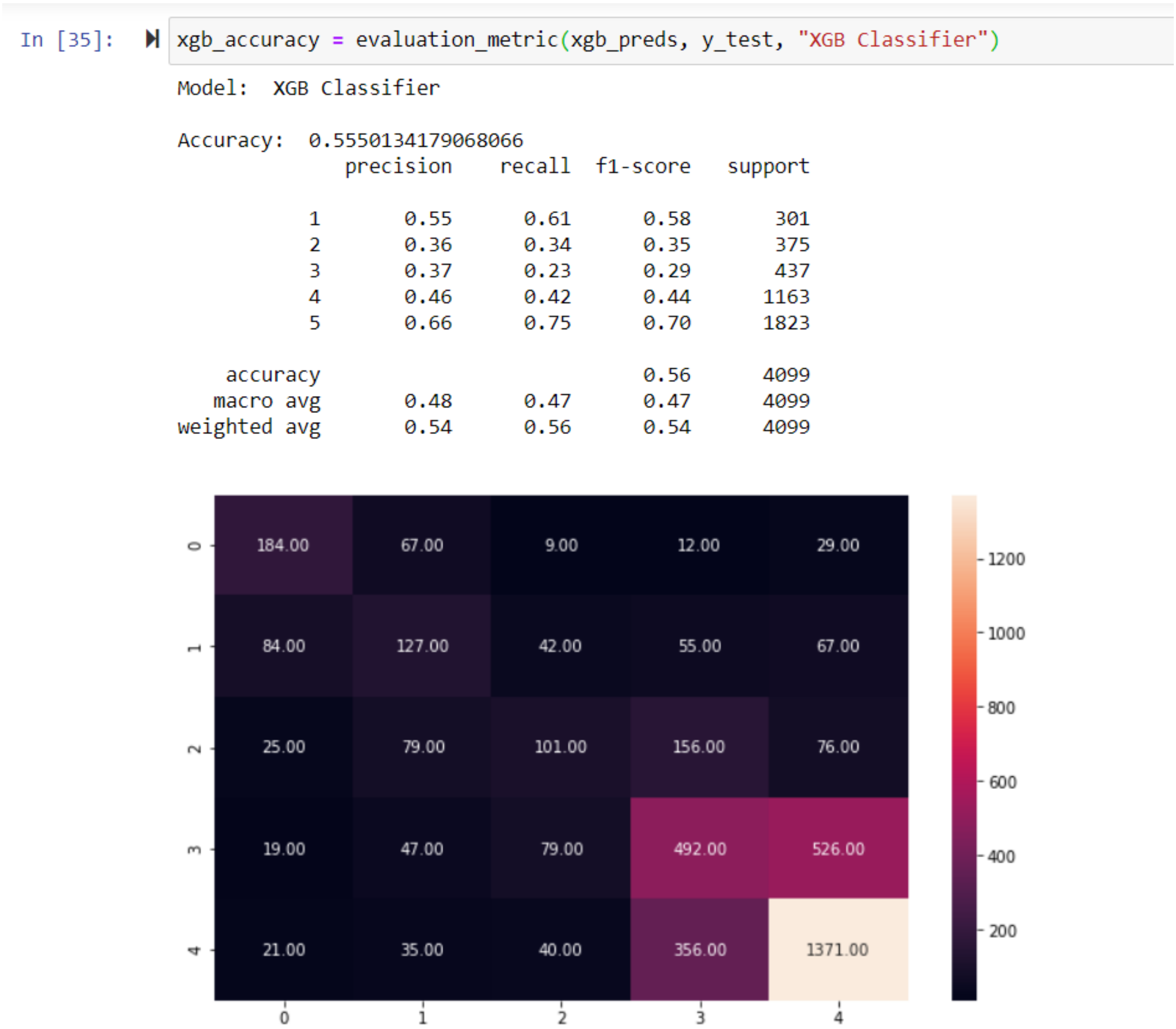


```
In [32]:  rf_accuracy = evaluation_metric(rf_preds, y_test, "Random Forest Classifier")
```

```
Model:  Random Forest Classifier

Accuracy:  0.5206147840936813
              precision    recall  f1-score   support

           1       0.59      0.53      0.56       301
           2       0.36      0.14      0.20       375
           3       0.23      0.11      0.14       437
           4       0.40      0.33      0.36      1163
           5       0.59      0.82      0.69      1823

    accuracy                           0.52      4099
   macro avg       0.43      0.39      0.39      4099
weighted avg       0.48      0.52      0.48      4099
```

	0	1	2	3	4
0	160.00	26.00	11.00	27.00	77.00
1	80.00	52.00	32.00	84.00	127.00
2	25.00	31.00	46.00	185.00	150.00
3	6.00	23.00	70.00	387.00	677.00
4	1.00	13.00	39.00	281.00	1489.00

Fig 9.4:Random forest accuracy and heat map

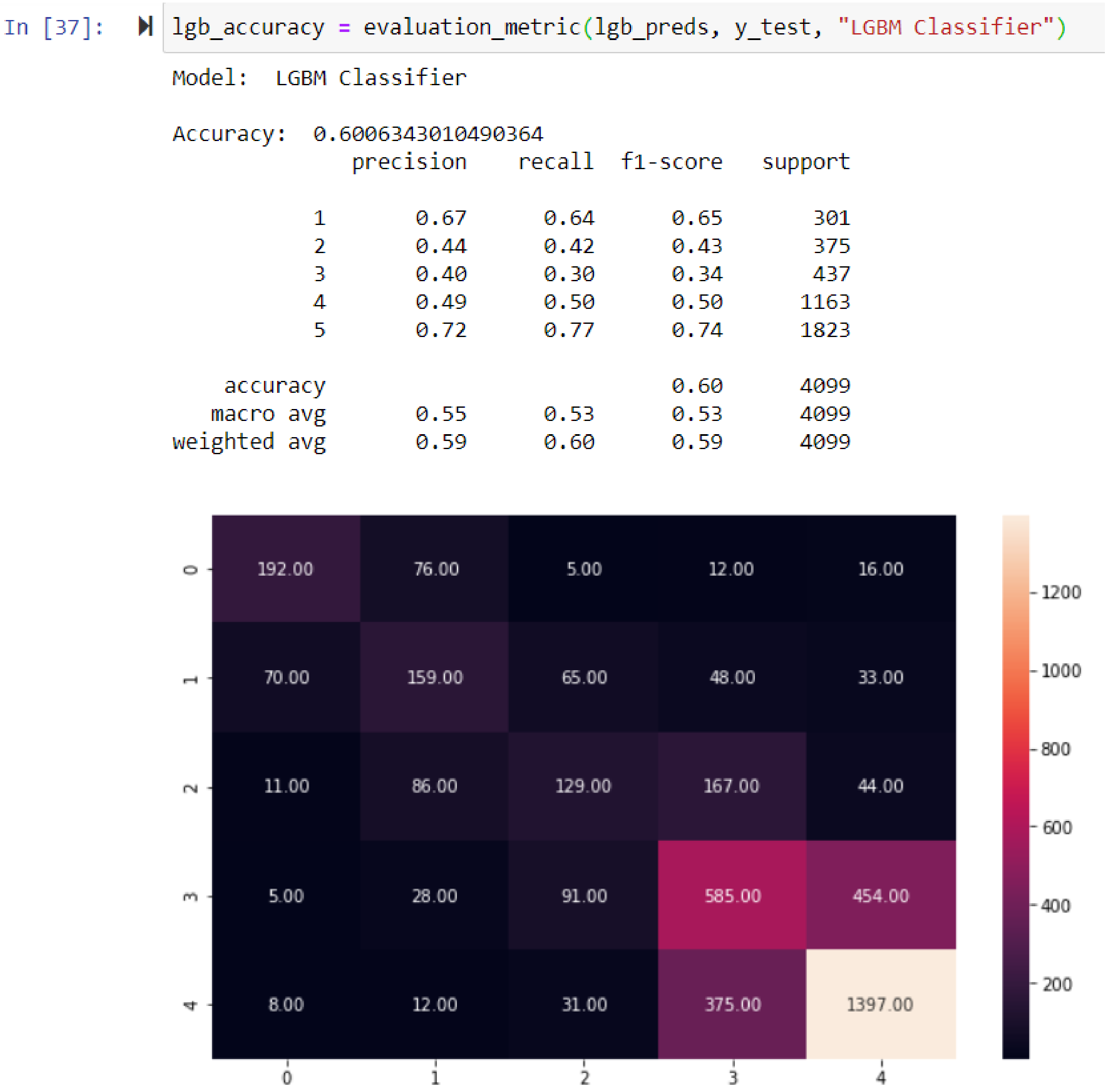


```
In [35]:  xgb_accuracy = evaluation_metric(xgb_preds, y_test, "XGB Classifier")
```

```
Model:  XGB Classifier

Accuracy:  0.5550134179068066
              precision    recall  f1-score   support

           1       0.55      0.61      0.58       301
           2       0.36      0.34      0.35       375
           3       0.37      0.23      0.29       437
           4       0.46      0.42      0.44      1163
           5       0.66      0.75      0.70      1823

    accuracy                           0.56      4099
   macro avg       0.48      0.47      0.47      4099
weighted avg       0.54      0.56      0.54      4099
```

	0	1	2	3	4
0	184.00	67.00	9.00	12.00	29.00
1	84.00	127.00	42.00	55.00	67.00
2	25.00	79.00	101.00	156.00	76.00
3	19.00	47.00	79.00	492.00	526.00
4	21.00	35.00	40.00	356.00	1371.00

Fig 9.5:XGB classifier accuracy and heat map

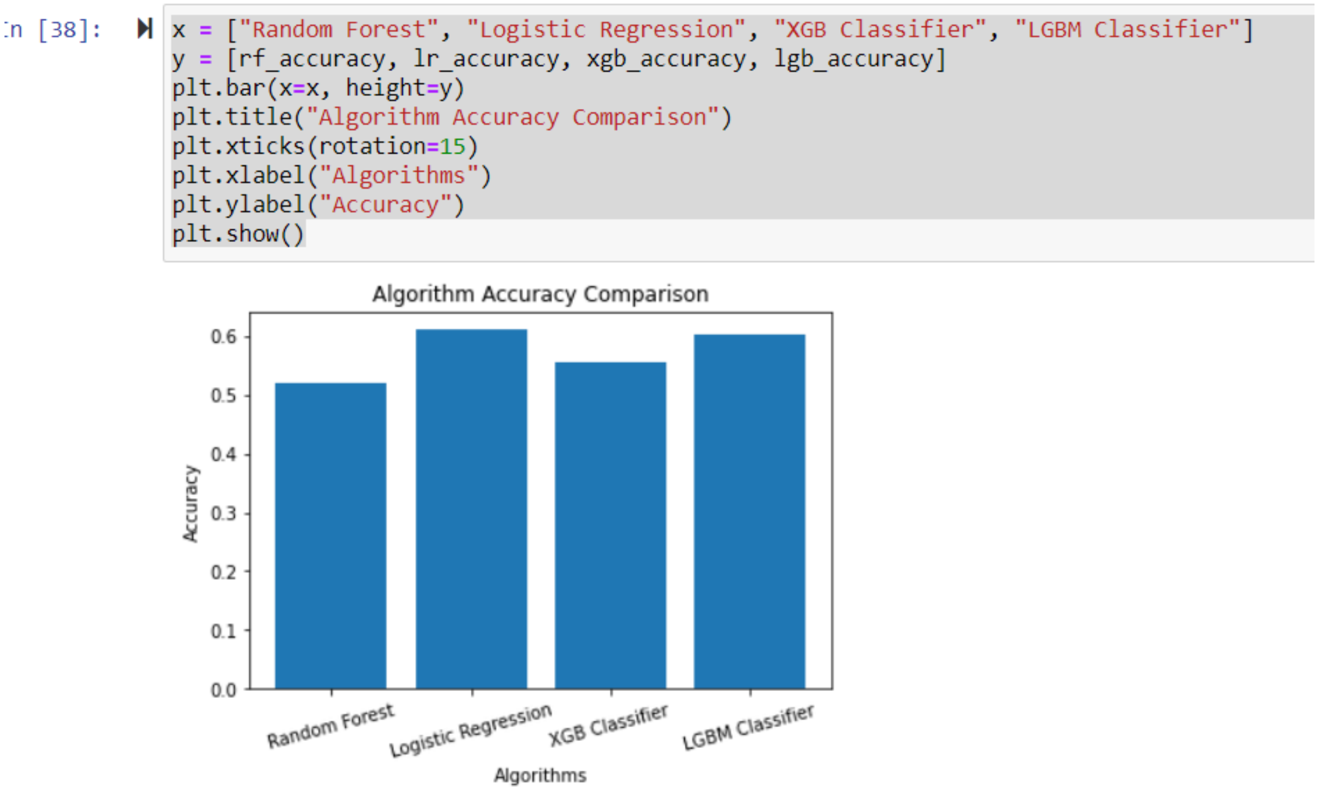


```
In [37]:  lgb_accuracy = evaluation_metric(lgb_preds, y_test, "LGBM Classifier")
```

```
Model:  LGBM Classifier

Accuracy:  0.6006343010490364
              precision    recall  f1-score   support

           1       0.67      0.64      0.65       301
           2       0.44      0.42      0.43       375
           3       0.40      0.30      0.34       437
           4       0.49      0.50      0.50      1163
           5       0.72      0.77      0.74      1823

    accuracy                           0.60      4099
   macro avg       0.55      0.53      0.53      4099
weighted avg       0.59      0.60      0.59      4099
```

	0	1	2	3	4
0	192.00	76.00	5.00	12.00	16.00
1	70.00	159.00	65.00	48.00	33.00
2	11.00	86.00	129.00	167.00	44.00
3	5.00	28.00	91.00	585.00	454.00
4	8.00	12.00	31.00	375.00	1397.00

Fig 9.6: LGBM classifier accuracy and heat map.



```
x = ["Random Forest", "Logistic Regression", "XGB Classifier", "LGBM Classifier"]
y = [rf_accuracy, lr_accuracy, xgb_accuracy, lgb_accuracy]
plt.bar(x=x, height=y)
plt.title("Algorithm Accuracy Comparison")
plt.xticks(rotation=15)
plt.xlabel("Algorithms")
plt.ylabel("Accuracy")
plt.show()
```

Fig 9.7:Algorithm accuracy comparison.