PHISHING SITE DETECTION

Sl. No.	Reg. No.	Student Name
1	18ETCS002068	M.C Dharshini
2	18ETCS002075	Mydur Manish Kumar
3	18ETCS002076	Naveen Kumar R C
4	18ETCS002095	Rayanagoud Nadagoudra
5	18ETCS002100	Rohith N

Supervisor: Ms. Pallavi R Kumar

JUNE-2022

B. Tech. in Computer Science and Engineering

FACULTY OF ENGINEERING AND TECHNOLOGY

M. S. RAMAIAH UNIVERSITY OF APPLIED SCIENCES

Bengaluru -560 054

FACULTY OF ENGINEERING AND TECHNOLOGY

FET

Certificate

This is to certify that the Project titled "Phishing Site Detection" is a bonafide work carried out in the Department of Computer Science and Engineering by Mydur Manish Kumar, Naveen Kumar R C, Rayanagoud Nadagoudra, Rohith N and M.C Dharshinibearing Register Number. 18ETCS002075, 18ETCS002076, 18ETCS002095, 18ETCS002100, 18ETCS002068 respectively in partial fulfilment of requirements for the award of B. Tech. Degree in Computer Science and Engineering of Ramaiah University of Applied Sciences.

JUNE – 2022

Ms. Pallavi R Kumar
Assistant Professor, Department of CSE

Dr T P Pushpavathi
Professor and Head – Dept. of CSE

Dr.Dilip Kumar Mahanty
Professor and Dean-FET

Declaration

PHISHING SITE DETECTION

The project work is submitted in partial fulfilment of academic requirements for the awardof **B.Tech**. Degree in the **Department of Computer Science and Engineering** of the Faculty of **Engineering and Technology** of Ramaiah University of Applied Sciences. The project report submitted herewith is a result of our own work and in conformance to the guidelines on plagiarism as laid out in the University Student Handbook. All sections of thetext and results which have been obtained from other sources are fully referenced. We understand that cheating and plagiarism constitute a breach of University regulations, hence this project report has been passed through plagiarism check and the report has been submitted to the supervisor.

Sl. No.	Reg. No.	Student Name
1	18ETCS002068	M.C Dharshini
2	18ETCS002075	Mydur Manish Kumar
3	18ETCS002076	Naveen Kumar R C
4	18ETCS002095	Rayanagoud Nadagoudra
5	18ETCS002100	Rohith N

Date: 3 June 2022

Acknowledgements

It is with extreme pleasure and pride that we present our B-Tech. dissertation titled "***Phishing Site Detection***". We would like to express our sincere thanks and gratitude to the following people, who stood by us throughout, helping us with much required inputs, guidance, knowledge and supported us.

We take great pleasure to express our sincere thanks and gratitude to academic project guide **Pallavi R Kumar** Asst. Professor Department of CSE, for his support, guidance andsuggestions throughout the project which is leading this project for the completion.

We express our sincere thanks to**, Dr Govind R Kadambi**, our respected Dean and to

Dr T P Pushpavathi, Head of Department of Computer Science and Engineering, for theirkind cooperation and support toward out dissertation, and to the management of Ramaiah University of Applied Science for their continued support. We are thankful to thestaff members of the Computer Science and Engineering, RUAS for giving us good supportand suggestion.

Lastly, we would like to thanks our parents and friends for their continued support, encouragement and motivation and God for paving our way of success in this object.

Cyber phishing is viewed as a theory of individual data in which phishers, otherwise called attackers, bait clients to give up delicate information, for example, qualifications, charge card and ledger data, monetary subtleties, and other social information. Phishing recognition is turning into a urgent exploration region, drawing in expanded center as the quantity of phishing assaults develops. Besides, in light of the fact that attackers are improving different procedures, identification has turned into an essential worry of engineers. Various phishing discovery plans has been incorporated into their engineering, like whitelist-, blacklist, content, visual comparability and URL-based overall. Each enjoys its singular benefits and downsides. In this overview report, we underscore on URL-based phishing discovery procedures, since we believe the URL to be a huge criterium in forestalling phishing assaults. In addition, looking at URL-based highlights can likewise energize quicker handling than different methodologies. In this work, we expect to comprehend the design of URL-based elements and looking over their assorted identification strategies and components. We then break down the presentation in view of the mixes of URL highlights on various datasets. At long last, we sum up our discoveries to advance better URL-based phishing discovery frameworks.

We have developed our project using website as a platform for all the users. This is an interactive and responsive website that will be used to detect whether a website is benign or phishing.

This website is made using different web designing language which include HTML, CSS and JavaScript. The website is created with an opinion such that people are not only able to distinguish between phishing or benign website, but also become aware of the mal-practices occurring in current world.

Keyword -Phishing, URL-based, Web Security, Features

List of Figures

1. Introduction

In this chapter, the project is introduced with its theme followed by its purpose. Then a literature survey is documented stating most popular existing applications under domainof current project and their key features. In the end, entire summary will be summarized.

1.1 Introduction

Phishing is a cyber-assault. The objective is to recover individual data of the beneficiary by causing them to accept that the message is something they need or need i.e., demand from the bank and so on. Phishing utilizes two phishing procedures malware and tricky based phishing. The assault can happen in two ways either by getting dubious email that prompted extortion webpage or by clients getting to joins that go straightforwardly to a phishing site.By and large, two methodologies are utilized in distinguishing the phishing locales. The first is based on blacklist. It works by contrasting the mentioned URL and those in that rundown however this approach has a disadvantage that is all there is to it can't distinguish another extortion site which has been made inside a small portion of second. The last option approach is heuristic based approach. In this methodology a few highlights are gathered from different site to characterize it into either phishing or real.

Expectation and avoidance of phishing assault is extremely significant stage towards shielding on the web exchange. The point is to foster a model to shield clients from phishing assaults. This should be possible utilizing special elements of phishing sites. The objective of our report is to foster a web application which informs the client when it distinguishes a phishing website.

Various types OF PHISHING ATTACKS

• Malware-Based Phishing: - It alludes to the execution of devilish programming on the client's PC. Malwares are encroached alongside a connection in the email, as the downloadable documents can follow the contributions from console.

• Deceptive Phishing: - Actual importance of phishing is secretarial taking utilizing direct correspondence yet these days the most normally utilized strategy is deceptive informing. The message shipped off the casualty worries about the need of confirmation of record subtleties, system disappointment makes it compulsory to reappear the subtleties of clients, counterfeit charges, troublesome changes in account, unforeseen free arrangements prompting quick activities, and a ton of more are being communicated to most extreme number of beneficiaries trusting

that the honest people might fall in their snare.

• System Reconfiguration:- Attacks might apply undesirable changes in the client's machine for devilish purposes. Representation: Websites which are referenced in for the most part utilized documents can be changed so that equivalent site is visited over and again.

• Hosts File Poisoning: - A URL is changed over into an IP address before it is communicated over the Internet.

• Information Shoplifting: - PCs without security might comprise of powerless data being put away on safeguarded servers.

A significant number of the machines are utilized to move toward such sort of servers for additional utilization.

• Pharming: - By utilizing this plan, interlopers might control an organization's space or host record so the requests for the office might make misleading correspondences with a manufactured site.

• Content-Injection Phishing: - Hackers control the items in a genuine destinations with misleading substance to mislead the client into surrendering their classified data to the programmer.

• Phishing through Search Engines: - Many undesirable ads of items and administrations are brought into the web crawlers offering items or administrations at a less expensive rate.

• Telephone Phishing: - Here, the person who does phishing utilizes sound calls to the client and try in controlling him.

• Malware Phishing: - It runs on the client's machine.

1.2 LITERATURE SURVEY

Similarly as phishing has different novel attributes, so do the recognition strategies and techniques. Be that as it may, phishing approaches can commonly be grouped into five classifications: whitelist-based, blacklist-, content-based, visual closeness based and URL-based. We list an outline of each methodology for a superior comprehension as follows:

1.2.1 Content-Based Approach

Zhang et al introduced an original methodology, purported CANTINA in 2007. Their work is based on Term Frequency - Inverse Document Frequency (TF-IDF) data recovery calculation used to identify phishing sites. Bar alone brought about a high

misleading positive rate because of restrictions on the quantity of web index results. This actually intends that as they increment the quantity of results, misleading positive rate will diminish while genuine positive rate continues as before, which isn't ideal. Subsequently, they utilized a few heuristics to diminish the bogus positive rate and further develop exactness. Their methodology accomplished an improved result contrasted with well-known enemy of phishing toolbars, accomplishing 97% genuine positive and 1% bogus positive rate. In 2011, Xiang et al further superior CANTINA, calling it CANTINA+, which is viewed as the most complete component rich methodology in satisfied based phishing discovery. It accomplished a superior 0.4% misleading positive rate and more than 92% genuine positive rate. Nonetheless, since the two methodologies use web indexes and outsider administrations, DNS compromising turned into a difficult danger

1.2.2. Blacklist-Based Approach

Internet browsers-like Google Safe Browsing - that protect against phishing assaults by refreshing a rundown of boycotted locales. In 2008, Sharifi et al proposed another dark ist generator strategy to settle the normal issues of keeping an exceptional rundown. Notwithstanding, since their proposed system depends on outsider administrations (like Google) for looking through space name to analyze top outcomes, it brings about terrible showing. Besides, blacklist approaches experience the significant issue of party time phishing assaults on the grounds that recently made phishing destinations are not in the rundown. PhishNet likewise predicts phishing assaults based on a blacklist plot. It utilizes five heuristics-top level space, IP address, index structure, question string and brand name-for mixes of blacklists to anticipate new phishing destinations. In spite of the fact that it can't recognize party time phishing locales, it accomplishes 95% genuine positive rate and 3% misleading positive rate over huge datasets.

1.2.3. Whitelist-Based Approach

Kang et al proposed a methodology based on white-recorded destinations in 2007. They played out a URL likeness check to recognize phishing destinations from in any case and a component contrasting and Domain Name System (DNS) question to defeat DNS pharming chases down issue for depending on DNS from past explores. In 2008, Cao et al.Also introduced a computerized individual white-list approach, in which the system keeps a client's past login and cautions when new access has happened. Despite the fact that whitelist-based strategies appear to be compelling for phishing recognition, there is an impediment on getting genuine locales generally on the web. A bountiful rundown of solid sites is important for a vigorous system with high precision; in any case, bogus positive rates increment because of an absence of white-recorded sites data, which is essentially difficult to gather all genuine locales on the planet.

.1.2.4 URL-Based Approach

M. Aburrous, M. A. Hossain, K. Dahal and F. Thabtah proposed and canny phishing recognition system for ebanking utilizing fluffy information mining in 2010. The trial was performed based on fluffy rationale with information mining algorithms. They showed how compelling URL-based approaches are for phishing location. By and large, URL-based strategies perform quicker than some other, including contentand visual-closeness based approaches. All the more significantly, they function admirably on party time phishing assaults, which are turning into a central issue in current enemy of phishing society. In forthcoming areas, we further examine subtleties of URL based recognition.

1.2.5.Visual-Similarity-Based Approach

Wenyin et al proposed a straightforward visual-similarity-based approach in 2005. Their system performed phishing recognition on three degrees of similarity grids; (I) block level similarity, (ii) format similarity and (iii) generally speaking style similarity. Notwithstanding, the most delegate work on visual similarity was subsequently introduced by Fu et al in 2006 utilizing the Earth Mover Distance (EMD). EMD was utilized to compute the marks of two pictures for visual similarity. In spite of the fact that their technique performed well in precision with 89% genuine positive and 0.71% bogus positive rates, the huge responsibility expected to handle two pictures was an exhibition disadvantage, contrasted with different methodologies. Chen et al presented a heuristic enemy of phishing system to display perceptual similarity. They utilized a strategic relapse algorithm for normalizing page content highlights. Albeit the proposed technique accomplished 100 percent genuine positive rate, it had 0.74% misleading positive rate, which could be moved along.

1.2.6. Other Approaches

An assortment of elective strategies are involved by scientists in phishing discovery. Such strategies incorporate heuristic, hybrid, AI, DNS based and others Additionally, a few overviews with respect to various plans are performed by scientists.

1.3. CONCLUSIONS

In this report, introduced with its theme followed by its purpose. Also ,technical report with respect to phishing site detection were surveyed in order to get the knowledge of the past work done in the domain. The main objective of the report is to detect the phishing site.

In this project, the whole theory related to the proposed project which includes the technical and resource aspect is explained. This section begins with a discussion of Planning Languages and Practical Framework, the IDEs on which the proposed project is built. Finally, the whole theory will be summarized in a nutshell.

2.1 BACKGROUND THEORY:

The background theory gives complete knowledge and understanding of different frameworks and technologies required to complete the project.

2.1.1 Hypertext Mark-up Language (HTML):

Hypertext is a text which is used for linking one text to another text. HTML has its own semantics. Hypertext mark-up language is a document layout language and it is used to create structure and appearance of the webpage.

HTML is a programming language basically used for describing web pages.

- HTML stands for Hyper Text Mark-up Language
- HTML is a mark-up language.
- A mark-up language is a set of mark-up tags.
- The tags describe document content.
- HTML document contains HTML tags and plain text.
- HTML documents are also called web pages.

ADVANTAGES:

- With the knowledge of HTML we can create a web site.
- HTML is a plain text so it is easy to edit.
- With the help of HTML we can optimize our websites, to increase speed and performance to yield best result.
- Once we understand the basic of HTML then other related programming languages like JavaScript, Angular and PHP will be easier to understand.

APPLICATIONS:

- Web pages development.
- Embedding images and videos.
- Navigating the internet.
- Game development.

2.1.2 Cascading Style Sheets (CSS):

Cascading Style Sheets (CSS) is a rule based language that applies styling to your

HTML element. We write CSS rules in elements, and modify properties of those elements such as color, background colour, width, border thickness, font size etc. CSS is a technology that helps us make our website look more attractive. It is data sent from the website to the browser. This data tells the browser some rule for how the site should be displayed.

ADVANTAGES:

- Better website speed.
- Easier to maintain.
- Consistent design.
- Time-saving.
- Better device compatibility.
- Flexible positioning of design element.

2.1.3 JavaScript (JS):

- JavaScript is one of the most popular and widely used programming language all over the world.
- JavaScript is an object based scripting language which is used to create client-side dynamic web pages.
- In 1995 when JavaScript was released for front-end purpose but now it is used for both front-end and back-end of web development also.
- 97% of websites are using JavaScript for their client-side programming language.
- To become a web developer it is must to learn JavaScript.

APPLICATIONS:

- **JavaScript** is used for web development.
- JavaScript frameworks are used for developing and building web applications.
- Through Node.js, server-side software can be written using JavaScript.
- With the help of Node.js developers can use JavaScript to create a web server.
- JavaScript and HTML5 together plays a major role in games development.
- Creating mobile applications is one of the most powerful applications of JavaScript.

2.1.4 Python:

- Uuncomplicated and robust programming language that delivers both the power and complexity of traditional compiled languages along with the ease-of-use of simpler scripting and interpreted languages.

Some of the features are:

- High-level, Object-oriented, Scalable, Extensible, Portable, Easy-to-learn, Easy-to-read, Easy-to-maintain, Robust, Effective as a Rapid Prototyping Tool, Memory Manager, Interpreted and (Byte-) Compiled

- Python is high-level, dynamically typed and interpreted.
- Some of the Python Applications are Django, Flask, Pyramid, PyGame, Tkinter, Tryton, TimPlayer, Cplay, Fandango, Raspberry Pi
- Some of the popular libaries for data sciences are Pandas, NumPy, Ipython, matplotlib

APPLICATIONS:

- Web Scraping Applications
- Business Applications
- CAD Applications
- Embedded Applications
- Audio and Video Applications
- Web Scraping Applications
- Business Applications
- CAD Applications
- Embedded Applications
- Audio and Video Applications

2.1.5 MACHINE LEARNING:

Three definitions of Machine Learning are offered.

1. Arthur Samuel (1959). Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed.
2. Tom Mitchell (1998) Well-posed Learning Problem: A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.
3. Machine Learning is an application of Artificial Intelligence (AI) which enables a program (software) to learn from the experiences and improve their self at a task without being explicitly programmed.

Why do we need Machine Learning?

- Machine Learning can automate many tasks, especially the ones that only humans can perform with their innate intelligence.
- businesses can automate routine tasks
- automating and quickly create models for data analysis
- Helps in creating models that can process and analyze large amounts of complex data to deliver accurate results.
- Image recognition, text generation, and many other use-cases are finding applications in the real world.

APPLICATIONS:

- **Augmentation**: Machine learning, which assists humans with their day-to-day tasks, personally or commercially without having complete control of the output. Such machine learning is used in different ways such as Virtual Assistant, Data analysis, software solutions.
- **Automation**: Machine learning, which works entirely autonomously in any field without the need for any human intervention.
- **Finance Industry** Machine learning is growing in popularity in the finance industry. Banks are mainly using ML to find patterns inside the data but also to prevent fraud.
- **Government organization** the government makes use of ML to manage public safety and utilities.
- **Healthcare industry** Healthcare was one of the first industry to use machine learning with image detection.
- **Marketing** Broad use of AI is done in marketing thanks to abundant access to data.
- Prediction: Machine learning can also be used in the prediction systems.
- Image recognition: Machine learning can be used for face detection in an image as well.
- Speech Recognition: it is the translation of spoken words into the text. It is used in voice searches and more.
- Medical diagnoses: ML is trained to recognize cancerous tissues.
- Financial industry and trading: companies use ML in fraud investigations and credit checks.

2.2 Conclusion:

In this Chapter, all background knowledge for completion of this project explained elaborately. Also, in each mentioned resource and technology, the reason for its use is also expressed. Languages like HTML, CSS, JS and its use in this project is explained.

3.Aim and Objectives

This chapter focuses on defined title and Aim of the project correctly and clearly. Later this chapter also includes the required objectives that needed to be fulfilled in order to complete this project. This is followed by method and methodologies that tabulates the procedure at will be followed in order to complete the objectives. This section then endswith a summary.

3.1 Title

Phishing Site Detection

3.2 Aim

The aim to develop a website which determines whether the given URL of the corresponding website is secure or not.

3.3 Objectives

The objectives of the proposed Project are listed below:

1. To carry out literature survey on phishing website detection approach
2. To Collect dataset containing phishing and legitimate websites from the open source platform.
3. To extract the required features from the URL database.
4. To Divide the dataset into training and testing sets.
5. To Run selected machine learning and deep neural network algorithm like multilayer perceptron, Random forest, Decision tree on the dataset.
6. To implement the trained model and detect whether the website is phishing or not
7. To document a descriptive report by integrating the result.

3.4 Method and Methodology

- **Loading the data :**

An assortment of site URLs for 10000+ sites. Each example has some site Parameters and a class label recognizing it as a phishing site(1) or not (-1)

The following are the highlights that we have extracted for location of phishing URLs.

IP address presence in URL: A large portion of the harmless website don't utilize IP address as a URL to download site page. Utilization of IP address in URL shows that assailant is attempting to take delicate data.If IP address is present in URL then the component is set to 1 else set to -1.

Host name length: If URL's length is more noteworthy than 25 then the component is set to 1 else to -1 the Average, because length of the harmless URLs is viewed as a 25.

Checking the @ symbol in URL: Phishers add unique image @ in the URL drives the program to disregard the going before the "@" image and the genuine location frequently follows the "@" image, If @ image present in URL then the element is set to 1 else set to -1.

Redirecting URL's: The presence of "//" inside the URL way implies that the client will be diverted to another site. If "//" present in URL way then highlight is set to 1 else to -1.

"-" separates prefix and suffix in domain name: The "-" is seldom utilized in real URLs. Phishers add (-) this symbol to the space name so clients feel that they are managing with a real site page. If domain name isolated by (-) image then highlight is set to 1 else to -1.

For instance: Actual website is http://www.msruas.com yet phisher can make another phony site like http://www.msr-u-as.com to befuddle the guiltless clients

Presence of HTTPS as part of domain: Phishers might add the "HTTPS" token to the space a piece of a URL to deceive clients. If HTTPS token present in URL then the component is set to 1 else to -1.

For instance: http://https-www-amezon-in items.com

Slash number in the URL: The quantity of slashs in harmless URLs is viewed as a 5, in the event that number of slices in URL is more prominent than 5, the component is set to 1 else to -1.

- **Familiarizing with data**

  In this progression, few dataframe strategies are utilized to investigate the data and its features .It is useful in checking in the event that there is any missing worth in dataset and to check if any outliners present in the dataset.

- **data visualizing**

  Hardly any plots and charts are shown to find how the information is circulated and the how elements are connected with one another.

- **Dividing the Data**

Training and testing stage is vital in setting up the best model, here we utilized 80 percentage of the data to train the model and 20 percentage of the data to test the model.

- **Building the model and training**

Supervised machine is one of the most usually utilized and effective kind of machine learning. Supervised learning is utilized at whatever point we need to foresee a specific result/mark from a given set of features , and we have models of features- label pair's, We fabricate an ML model from these features- label pair's, which include our training set. We want to make exact expectations for new, never-before-seen information.

There are two significant kinds of regulated machine learning problems, called Classification and regression. Our informational collection goes under regression problem, as the prediction is for the continuous number.

Some of the supervised machine learning model we have used to train are:

- k-Nearest Neighbours
- Multilayer Perceptron's
- Logistic Regression
- Decision Tree
- Random Forest
- Gradient Boosting
- Catboast
- Xgboost
- Naive Bayes
- Support Vector Classifier

The metrics considered to evaluate the model performance are Accuracy & F1 score.

- **Comparison of models**

To compare machine learning algorithms it is important to check which model is best performed here to compare the performance of the model, a data framework is developed.

Sections of this data framework are records made to store model outcomes

3.5 Conclusion

This particular chapter focused on defined title and Aim of the project correctly. Later this topic included required objectives that were required to be fulfilled to complete this project. Phishing site detection are documented successfully in this section followed by method and methodologies which is successfully tabulated in order to know the steps used in completing the objectives including resources used.

4. Problem Solving

In this section, the actual dissection of project is done and each module is built piece by piece in order to complete the project. In design section, the application has been built in accordance with functional requirements based on block diagram. In implementation section, snips of important code is displayed with their explanation given below. In testingsection, all functional requirements are tested and the result is analysed resulting in statusof test condition. In the ending section, performance analysis is done on various factors such as battery, booting time and time requirement for major operation.

4.1 Design

Design is necessary when it comes to development since it acts as a blueprint for entire process from requirement making to finished (final product). Hence, in this section designs specific to this project like block diagram are attached.

4.1.1 Block Diagram

- Gathering the data (Creating the dataset):

  For training and testing purpose.

- Researching the model which will best for the type of data.
- Data pre-processing
- Training and testing the build model
- Evaluation for accuracy.

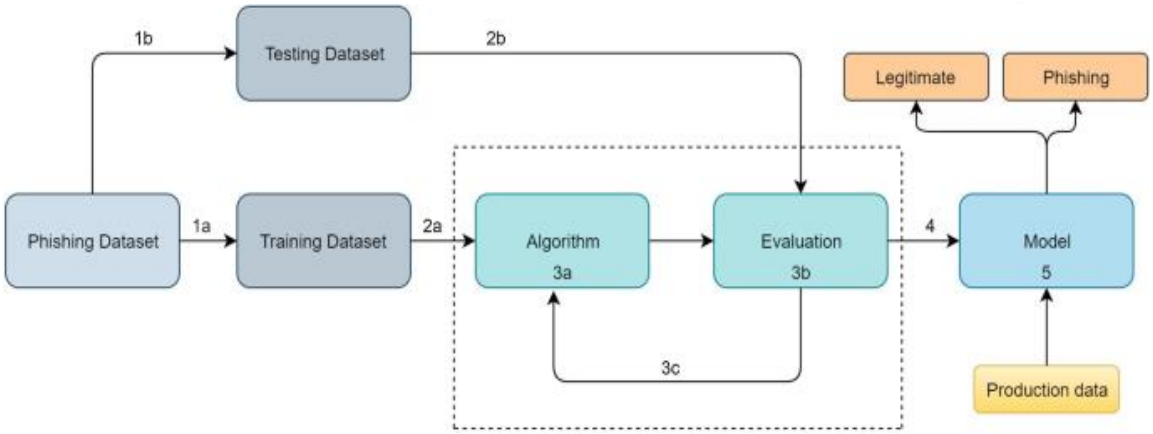


Figure 1 represent Block diagram.

4.2 Implementation

In the implementation, requirements are written in the coding language and transformedinto computer programs which are called software.

```
In [14]: # Creating holders to store the model performance results
         ML_Model = []
         accuracy = []
         f1_score = []
         recall = []
         precision = []

         #function to call for storing the results
         def storeResults(model, a,b,c,d):
           ML_Model.append(model)
           accuracy.append(round(a, 3))
           f1_score.append(round(b, 3))
           recall.append(round(c, 3))
           precision.append(round(d, 3))
```

Logistic Regression

It will work under principle of simple linear regression and it is used for classification mechanism problems in real time applications.

Based on the requirements we are classify the classification problems are

★ Binary classification

★ Multi-class classification

The logistics regression always produce result in binary format which is used to predict the outcomes of a categorical dependent variable in using the independent variable.

It gives the probabilistic values between 0 and 1.

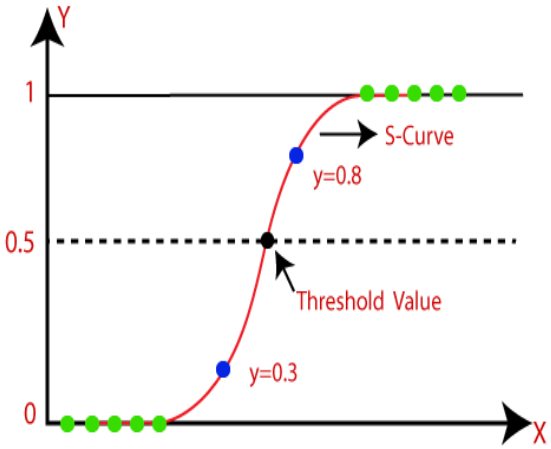


Figure 2 logistic function graph

Assumptions for logistic regression:

- The given dependent variable must be in categorical nature.
- Independent variable shouldn't have multi-collinearity.

In [15]:
```
# Linear regression model
from sklearn.linear_model import LogisticRegression
#from sklearn.pipeline import Pipeline

# instantiate the model
log = LogisticRegression()

# fit the model
log.fit(X_train,y_train)
```

Out[15]:
```
LogisticRegression()
```

In [16]:
```
#predicting the target value from the model for the samples

y_train_log = log.predict(X_train)
y_test_log = log.predict(X_test)
```

In [17]:
```
#computing the accuracy, f1_score, Recall, precision of the model performance

acc_train_log = metrics.accuracy_score(y_train,y_train_log)
acc_test_log = metrics.accuracy_score(y_test,y_test_log)
print("Logistic Regression : Accuracy on training Data: {:.3f}".format(acc_train_log))
print("Logistic Regression : Accuracy on test Data: {:.3f}".format(acc_test_log))
print()

f1_score_train_log = metrics.f1_score(y_train,y_train_log)
f1_score_test_log = metrics.f1_score(y_test,y_test_log)
print("Logistic Regression : f1_score on training Data: {:.3f}".format(f1_score_train_log))
print("Logistic Regression : f1_score on test Data: {:.3f}".format(f1_score_test_log))
print()

recall_score_train_log = metrics.recall_score(y_train,y_train_log)
recall_score_test_log = metrics.recall_score(y_test,y_test_log)
print("Logistic Regression : Recall on training Data: {:.3f}".format(recall_score_train_log))
print("Logistic Regression : Recall on test Data: {:.3f}".format(recall_score_test_log))
print()

precision_score_train_log = metrics.precision_score(y_train,y_train_log)
precision_score_test_log = metrics.precision_score(y_test,y_test_log)
print("Logistic Regression : precision on training Data: {:.3f}".format(precision_score_train_log))
print("Logistic Regression : precision on test Data: {:.3f}".format(precision_score_test_log))
```

```
Logistic Regression : Accuracy on training Data: 0.927
Logistic Regression : Accuracy on test Data: 0.934

Logistic Regression : f1_score on training Data: 0.935
Logistic Regression : f1_score on test Data: 0.941

Logistic Regression : Recall on training Data: 0.943
Logistic Regression : Recall on test Data: 0.953

Logistic Regression : precision on training Data: 0.927
Logistic Regression : precision on test Data: 0.930
```

In [18]:
```
#computing the classification report of the model

print(metrics.classification_report(y_test, y_test_log))
```

```
              precision    recall  f1-score   support

          -1       0.94      0.91      0.92       976
           1       0.93      0.95      0.94      1235

    accuracy                           0.93      2211
   macro avg       0.93      0.93      0.93      2211
weighted avg       0.93      0.93      0.93      2211
```

In [19]:
```
#storing the results. The below mentioned order of parameter passing is important.

storeResults('Logistic Regression',acc_test_log,f1_score_test_log,
             recall_score_train_log,precision_score_train_log)
```

K-Nearest Neighbors:

K-Nearest Neighbor is a supervised Machine Learning technique. This algorithm classify the new data based on the similarity measure. It is LAZY Algorithm for to memorize the process.

Algorithm is:

S1: loading the data from csv file

S2: initialize the k (i.e. k with some number that should be an odd number)

S3: for each sample in training data:

S3.1: calculate the distance between query point and current data point.

S3.2: Add the distance and the index of the example to an order collection.

S4: Sort (sorting the collection of the distances and indices in ascending order.)

S5: pick the K-entries from sorted collection.

S6: get the labels of selected entries

S7: if it is regression problem

Return the mean of k-labels

S8: if is classification problem:

Return the mode of k-labels.

In [20]:
```
# K-Nearest Neighbors Classifier model
from sklearn.neighbors import KNeighborsClassifier

# instantiate the model
knn = KNeighborsClassifier(n_neighbors=1)

# fit the model
knn.fit(X_train,y_train)
```

Out[20]:
```
KNeighborsClassifier(n_neighbors=1)
```

In [21]:
```
#predicting the target value from the model for the samples
y_train_knn = knn.predict(X_train)
y_test_knn = knn.predict(X_test)
```

In [22]:
```
#computing the accuracy,f1_score,Recall,precision of the model performance

acc_train_knn = metrics.accuracy_score(y_train,y_train_knn)
acc_test_knn = metrics.accuracy_score(y_test,y_test_knn)
print("K-Nearest Neighbors : Accuracy on training Data: {:.3f}".format(acc_train_knn))
print("K-Nearest Neighbors : Accuracy on test Data: {:.3f}".format(acc_test_knn))
print()

f1_score_train_knn = metrics.f1_score(y_train,y_train_knn)
f1_score_test_knn = metrics.f1_score(y_test,y_test_knn)
print("K-Nearest Neighbors : f1_score on training Data: {:.3f}".format(f1_score_train_knn))
print("K-Nearest Neighbors : f1_score on test Data: {:.3f}".format(f1_score_test_knn))
print()

recall_score_train_knn = metrics.recall_score(y_train,y_train_knn)
recall_score_test_knn = metrics.recall_score(y_test,y_test_knn)
print("K-Nearest Neighborsn : Recall on training Data: {:.3f}".format(recall_score_train_knn))
print("Logistic Regression : Recall on test Data: {:.3f}".format(recall_score_test_knn))
print()

precision_score_train_knn = metrics.precision_score(y_train,y_train_knn)
precision_score_test_knn = metrics.precision_score(y_test,y_test_knn)
print("K-Nearest Neighbors : precision on training Data: {:.3f}".format(precision_score_train_knn))
print("K-Nearest Neighbors : precision on test Data: {:.3f}".format(precision_score_test_knn))
```

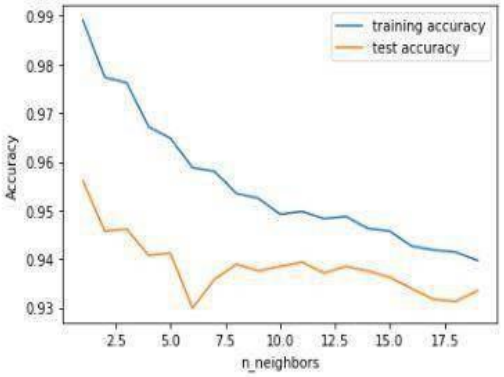
```
K-Nearest Neighbors : Accuracy on training Data: 0.989
K-Nearest Neighbors : Accuracy on test Data: 0.956
```

```
K-Nearest Neighbors : f1_score on training Data: 0.990
K-Nearest Neighbors : f1_score on test Data: 0.961

K-Nearest Neighborsn : Recall on training Data: 0.991
Logistic Regression : Recall on test Data: 0.962

K-Nearest Neighbors : precision on training Data: 0.989
K-Nearest Neighbors : precision on test Data: 0.960
```

In [23]:
```
#computing the classification report of the model

print(metrics.classification_report(y_test, y_test_knn))
```

```
              precision    recall  f1-score   support

          -1       0.95      0.95      0.95       976
           1       0.96      0.96      0.96      1235

    accuracy                           0.96      2211
   macro avg       0.96      0.96      0.96      2211
weighted avg       0.96      0.96      0.96      2211
```

In [24]:
```
training_accuracy = []
test_accuracy = []
# try max_depth from 1 to 20
depth = range(1,20)
for n in depth:
    knn = KNeighborsClassifier(n_neighbors=n)

    knn.fit(X_train, y_train)
    # record training set accuracy
    training_accuracy.append(knn.score(X_train, y_train))
    # record generalization accuracy
    test_accuracy.append(knn.score(X_test, y_test))


#plotting the training & testing accuracy for n_estimators from 1 to 20
plt.plot(depth, training_accuracy, label="training accuracy")
plt.plot(depth, test_accuracy, label="test accuracy")
plt.ylabel("Accuracy")
plt.xlabel("n_neighbors")
plt.legend();
```



In [25]:
```
#storing the results. The below mentioned order of parameter passing is important.

storeResults('K-Nearest Neighbors',acc_test_knn,f1_score_test_knn,
             recall_score_train_knn,precision_score_train_knn)
```

Support Vector Classifier:

Support Vector classifier or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems.
Support vector classifier for linearly separable binary sets. The goal is to design a hyperplane that classifiers all training vectors in two classes. The best choice will be the hyperplane that leaves the maximum margin from both classes.

In [26]:
```
# Support Vector Classifier model
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV

# defining parameter range
param_grid = {'gamma': [0.1],'kernel': ['rbf','linear']}

svc = GridSearchCV(SVC(), param_grid)

# fitting the model for grid search
svc.fit(X_train, y_train)
```

Out[26]:
```
GridSearchCV(estimator=SVC(),
             param_grid={'gamma': [0.1], 'kernel': ['rbf', 'linear']})
```

In [27]:
```
#predicting the target value from the model for the samples
y_train_svc = svc.predict(X_train)
y_test_svc = svc.predict(X_test)
```

In [28]:
```
#computing the accuracy, f1_score, Recall, precision of the model performance

acc_train_svc = metrics.accuracy_score(y_train,y_train_svc)
acc_test_svc = metrics.accuracy_score(y_test,y_test_svc)
print("Support Vector Machine : Accuracy on training Data: {:.3f}".format(acc_train_svc))
print("Support Vector Machine : Accuracy on test Data: {:.3f}".format(acc_test_svc))
print()

f1_score_train_svc = metrics.f1_score(y_train,y_train_svc)
f1_score_test_svc = metrics.f1_score(y_test,y_test_svc)
print("Support Vector Machine : f1_score on training Data: {:.3f}".format(f1_score_train_svc))
print("Support Vector Machine : f1_score on test Data: {:.3f}".format(f1_score_test_svc))
print()

recall_score_train_svc = metrics.recall_score(y_train,y_train_svc)
recall_score_test_svc = metrics.recall_score(y_test,y_test_svc)
print("Support Vector Machine : Recall on training Data: {:.3f}".format(recall_score_train_svc))
print("Support Vector Machine : Recall on test Data: {:.3f}".format(recall_score_test_svc))
print()

precision_score_train_svc = metrics.precision_score(y_train,y_train_svc)
precision_score_test_svc = metrics.precision_score(y_test,y_test_svc)
print("Support Vector Machine : precision on training Data: {:.3f}".format(precision_score_train_svc))
print("Support Vector Machine : precision on test Data: {:.3f}".format(precision_score_test_svc))
```

```
Support Vector Machine : Accuracy on training Data: 0.969
Support Vector Machine : Accuracy on test Data: 0.964

Support Vector Machine : f1_score on training Data: 0.973
Support Vector Machine : f1_score on test Data: 0.968

Support Vector Machine : Recall on training Data: 0.980
Support Vector Machine : Recall on test Data: 0.980

Support Vector Machine : precision on training Data: 0.965
Support Vector Machine : precision on test Data: 0.957
```

In [29]:
```
#computing the classification report of the model

print(metrics.classification_report(y_test, y_test_svc))
```

```
              precision    recall  f1-score   support

          -1       0.97      0.94      0.96       976
           1       0.96      0.98      0.97      1235

    accuracy                           0.96      2211
   macro avg       0.97      0.96      0.96      2211
weighted avg       0.96      0.96      0.96      2211
```

In [30]:
```
#storing the results. The below mentioned order of parameter passing is important.

storeResults('Support Vector Machine',acc_test_svc,f1_score_test_svc,
             recall_score_train_svc,precision_score_train_svc)
```

Naive Bayes:

Naïve Bayes algorithm works on Bayes theorem based on the independent assumptions between the predictors and this assumption is called class conditional independence. This model is easy to build without complicated iterative parameters estimation for large datasets. It is particular done for to reduce complication in iterative parameter estimations for very large data sets. And it is supervised Machine learning Technique for to solve the real-world classification problems.

Examples are text classification and image classification and this includes the high dimensional training dataset for train the model. And it will help in quick predictions.

In [31]:
```
# Naive Bayes Classifier Model
from sklearn.naive_bayes import GaussianNB
from sklearn.pipeline import Pipeline

# instantiate the model
nb=  GaussianNB()

# fit the model
nb.fit(X_train,y_train)
```

Out[31]:
```
GaussianNB()
```

In [32]:
```
#predicting the target value from the model for the samples
y_train_nb = nb.predict(X_train)
y_test_nb = nb.predict(X_test)
```

In [33]:
```
#computing the accuracy, f1_score, Recall, precision of the model performance

acc_train_nb = metrics.accuracy_score(y_train,y_train_nb)
acc_test_nb = metrics.accuracy_score(y_test,y_test_nb)
print("Naive Bayes Classifier : Accuracy on training Data: {:.3f}".format(acc_train_nb))
print("Naive Bayes Classifier : Accuracy on test Data: {:.3f}".format(acc_test_nb))
print()

f1_score_train_nb = metrics.f1_score(y_train,y_train_nb)
f1_score_test_nb = metrics.f1_score(y_test,y_test_nb)
print("Naive Bayes Classifier : f1_score on training Data: {:.3f}".format(f1_score_train_nb))
print("Naive Bayes Classifier : f1_score on test Data: {:.3f}".format(f1_score_test_nb))
print()

recall_score_train_nb = metrics.recall_score(y_train,y_train_nb)
recall_score_test_nb = metrics.recall_score(y_test,y_test_nb)
print("Naive Bayes Classifier : Recall on training Data: {:.3f}".format(recall_score_train_nb))
print("Naive Bayes Classifier : Recall on test Data: {:.3f}".format(recall_score_test_nb))
print()

precision_score_train_nb = metrics.precision_score(y_train,y_train_nb)
precision_score_test_nb = metrics.precision_score(y_test,y_test_nb)
print("Naive Bayes Classifier : precision on training Data: {:.3f}".format(precision_score_train_nb))
print("Naive Bayes Classifier : precision on test Data: {:.3f}".format(precision_score_test_nb))
```

```
Naive Bayes Classifier : Accuracy on training Data: 0.605
Naive Bayes Classifier : Accuracy on test Data: 0.605

Naive Bayes Classifier : f1_score on training Data: 0.451
Naive Bayes Classifier : f1_score on test Data: 0.454

Naive Bayes Classifier : Recall on training Data: 0.292
Naive Bayes Classifier : Recall on test Data: 0.294

Naive Bayes Classifier : precision on training Data: 0.997
Naive Bayes Classifier : precision on test Data: 0.995
```

In [34]:
```
#computing the classification report of the model

print(metrics.classification_report(y_test, y_test_svc))
```

```
              precision    recall  f1-score   support

          -1       0.97      0.94      0.96       976
           1       0.96      0.98      0.97      1235

    accuracy                           0.96      2211
   macro avg       0.97      0.96      0.96      2211
weighted avg       0.96      0.96      0.96      2211
```

In [35]:
```
#storing the results. The below mentioned order of parameter passing is important.

storeResults('Naive Bayes Classifier',acc_test_nb,f1_score_test_nb,
             recall_score_train_nb,precision_score_train_nb)
```

Decision Tree:

The Decision tree generates the regression or classification models based on the requirements in the form of tree data structure. it is supervise Machine learning technique to divides the given dataset into smaller and smaller subsets parallel at the same time an association decision tree developed incrementally. The final result is tree having leaf nodes and decision nodes. Where decision nodes has two or more branches and leaf node represents a problem is classification or regression. Topmost decision node in a final tree which correspond to the best predictor which is called as root node of the final tree. Branches represent the decisions rules between the nodes and it will handle the both categorical and numerical data. But the most preferred for classification problems compared to regression problems.

In [36]:
```
# Decision Tree Classifier model
from sklearn.tree import DecisionTreeClassifier

# instantiate the model
tree = DecisionTreeClassifier(max_depth=30)

# fit the model
tree.fit(X_train, y_train)
```

Out[36]:
```
DecisionTreeClassifier(max_depth=30)
```

In [37]:
```
#predicting the target value from the model for the samples

y_train_tree = tree.predict(X_train)
y_test_tree = tree.predict(X_test)
```

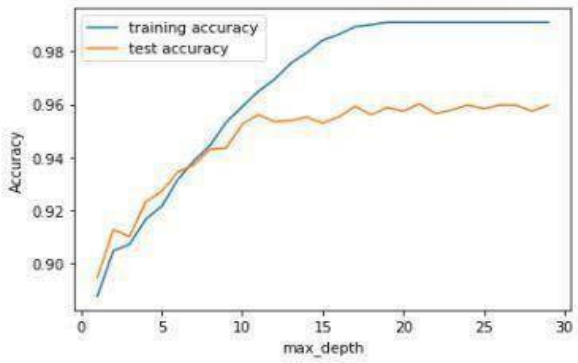
In [38]:
```
#computing the accuracy, f1_score, Recall, precision of the model performance

acc_train_tree = metrics.accuracy_score(y_train,y_train_tree)
acc_test_tree = metrics.accuracy_score(y_test,y_test_tree)
print("Decision Tree : Accuracy on training Data: {:.3f}".format(acc_train_tree))
print("Decision Tree : Accuracy on test Data: {:.3f}".format(acc_test_tree))
print()

f1_score_train_tree = metrics.f1_score(y_train,y_train_tree)
f1_score_test_tree = metrics.f1_score(y_test,y_test_tree)
print("Decision Tree : f1_score on training Data: {:.3f}".format(f1_score_train_tree))
print("Decision Tree : f1_score on test Data: {:.3f}".format(f1_score_test_tree))
print()

recall_score_train_tree = metrics.recall_score(y_train,y_train_tree)
recall_score_test_tree = metrics.recall_score(y_test,y_test_tree)
print("Decision Tree : Recall on training Data: {:.3f}".format(recall_score_train_tree))
print("Decision Tree : Recall on test Data: {:.3f}".format(recall_score_test_tree))
print()

precision_score_train_tree = metrics.precision_score(y_train,y_train_tree)
precision_score_test_tree = metrics.precision_score(y_test,y_test_tree)
print("Decision Tree : precision on training Data: {:.3f}".format(precision_score_train_tree))
print("Decision Tree : precision on test Data: {:.3f}".format(precision_score_test_tree))
```

```
Decision Tree : Accuracy on training Data: 0.991
Decision Tree : Accuracy on test Data: 0.960

Decision Tree : f1_score on training Data: 0.992
Decision Tree : f1_score on test Data: 0.964
```

```
print(metrics.classification_report(y_test, y_test_tree))
```

	precision	recall	f1-score	support
-1	0.95	0.96	0.95	976
1	0.97	0.96	0.96	1235
accuracy			0.96	2211
macro avg	0.96	0.96	0.96	2211
weighted avg	0.96	0.96	0.96	2211

In [40]:
```
training_accuracy = []
test_accuracy = []
# try max_depth from 1 to 30
depth = range(1,30)
for n in depth:
    tree_test = DecisionTreeClassifier(max_depth=n)

    tree_test.fit(X_train, y_train)
    # record training set accuracy
    training_accuracy.append(tree_test.score(X_train, y_train))
    # record generalization accuracy
    test_accuracy.append(tree_test.score(X_test, y_test))


#plotting the training & testing accuracy for max_depth from 1 to 30
plt.plot(depth, training_accuracy, label="training accuracy")
plt.plot(depth, test_accuracy, label="test accuracy")
plt.ylabel("Accuracy")
plt.xlabel("max_depth")
plt.legend();
```



Random Forest

It is a collection of multiple random decision trees and its much less sensitive to the training data.

The first step is to build new datasets from original data, we are going to randomly select rows from the original data to build our new datasets and every dataset will contain the same number of rows as the original one. We will randomly select a subset of features for each tree and use only them for training. We will pass this data point through each tree one by one and note down the predictions. After we need to combine all the predictions, as it's a classification problem we will take the majority voting. In random forest we first perform bootstrapping then aggregation. We have used two random processes bootstrapping and random forest selection. Bootstrapping ensures that we are not using the same data for every tree so in a way it helps over model to be less sensitive to the original training data. The random feature selection helps to reduce the correlation between the trees.

In [42]:
```
# Random Forest Classifier Model
from sklearn.ensemble import RandomForestClassifier

# instantiate the model
forest = RandomForestClassifier(n_estimators=10)

# fit the model
forest.fit(X_train,y_train)
```

Out[42]:
```
RandomForestClassifier(n_estimators=10)
```

In [43]:
```
#predicting the target value from the model for the samples
y_train_forest = forest.predict(X_train)
y_test_forest = forest.predict(X_test)
```

In [44]:
```
#computing the accuracy, f1_score, Recall, precision of the model performance

acc_train_forest = metrics.accuracy_score(y_train,y_train_forest)
acc_test_forest = metrics.accuracy_score(y_test,y_test_forest)
print("Random Forest : Accuracy on training Data: {:.3f}".format(acc_train_forest))
print("Random Forest : Accuracy on test Data: {:.3f}".format(acc_test_forest))
print()

f1_score_train_forest = metrics.f1_score(y_train,y_train_forest)
f1_score_test_forest = metrics.f1_score(y_test,y_test_forest)
print("Random Forest : f1_score on training Data: {:.3f}".format(f1_score_train_forest))
print("Random Forest : f1_score on test Data: {:.3f}".format(f1_score_test_forest))
print()

recall_score_train_forest = metrics.recall_score(y_train,y_train_forest)
recall_score_test_forest = metrics.recall_score(y_test,y_test_forest)
print("Random Forest : Recall on training Data: {:.3f}".format(recall_score_train_forest))
print("Random Forest : Recall on test Data: {:.3f}".format(recall_score_test_forest))
print()

precision_score_train_forest = metrics.precision_score(y_train,y_train_forest)
precision_score_test_forest = metrics.precision_score(y_test,y_test_tree)
print("Random Forest : precision on training Data: {:.3f}".format(precision_score_train_forest))
print("Random Forest : precision on test Data: {:.3f}".format(precision_score_test_forest))
```

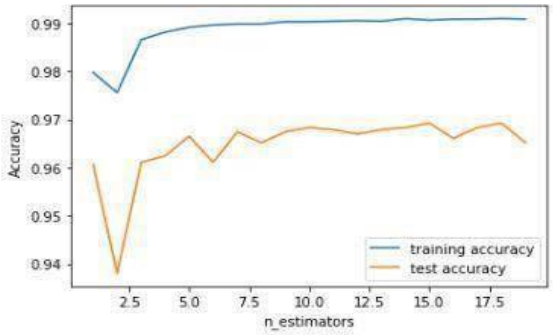
```
Random Forest : Accuracy on training Data: 0.991
Random Forest : Accuracy on test Data: 0.967

Random Forest : f1_score on training Data: 0.992
Random Forest : f1_score on test Data: 0.971
```

```
print(metrics.classification_report(y_test, y_test_forest))
```

```
              precision    recall  f1-score   support

          -1       0.97      0.96      0.96       976
           1       0.97      0.97      0.97      1235

    accuracy                           0.97      2211
   macro avg       0.97      0.97      0.97      2211
weighted avg       0.97      0.97      0.97      2211
```

In [47]:

```
training_accuracy = []
test_accuracy = []
# try max_depth from 1 to 20
depth = range(1,20)
for n in depth:
    forest_test =  RandomForestClassifier(n_estimators=n)

    forest_test.fit(X_train, y_train)
    # record training set accuracy
    training_accuracy.append(forest_test.score(X_train, y_train))
    # record generalization accuracy
    test_accuracy.append(forest_test.score(X_test, y_test))


#plotting the training & testing accuracy for n_estimators from 1 to 20
plt.figure(figsize=None)
plt.plot(depth, training_accuracy, label="training accuracy")
plt.plot(depth, test_accuracy, label="test accuracy")
plt.ylabel("Accuracy")
plt.xlabel("n_estimators")
plt.legend();
```



Gradient Boosting Classifier

In [49]:
```
# Gradient Boosting Classifier Model
from sklearn.ensemble import GradientBoostingClassifier

# instantiate the model
gbc = GradientBoostingClassifier(max_depth=4,learning_rate=0.7)

# fit the model
gbc.fit(X_train,y_train)
```

Out[49]:
```
GradientBoostingClassifier(learning_rate=0.7, max_depth=4)
```

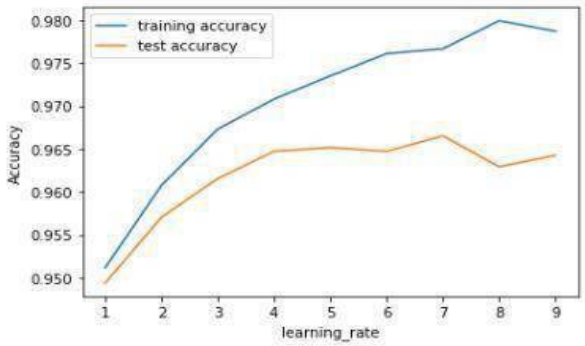
In [50]:
```
#predicting the target value from the model for the samples
y_train_gbc = gbc.predict(X_train)
y_test_gbc = gbc.predict(X_test)
```

In [51]:
```
#computing the accuracy, f1_score, Recall, precision of the model performance

acc_train_gbc = metrics.accuracy_score(y_train,y_train_gbc)
acc_test_gbc = metrics.accuracy_score(y_test,y_test_gbc)
print("Gradient Boosting Classifier : Accuracy on training Data: {:.3f}".format(acc_train_gbc))
print("Gradient Boosting Classifier : Accuracy on test Data: {:.3f}".format(acc_test_gbc))
print()

f1_score_train_gbc = metrics.f1_score(y_train,y_train_gbc)
f1_score_test_gbc = metrics.f1_score(y_test,y_test_gbc)
print("Gradient Boosting Classifier : f1_score on training Data: {:.3f}".format(f1_score_train_gbc))
print("Gradient Boosting Classifier : f1_score on test Data: {:.3f}".format(f1_score_test_gbc))
print()

recall_score_train_gbc = metrics.recall_score(y_train,y_train_gbc)
recall_score_test_gbc =  metrics.recall_score(y_test,y_test_gbc)
print("Gradient Boosting Classifier : Recall on training Data: {:.3f}".format(recall_score_train_gbc))
print("Gradient Boosting Classifier : Recall on test Data: {:.3f}".format(recall_score_test_gbc))
print()

precision_score_train_gbc = metrics.precision_score(y_train,y_train_gbc)
precision_score_test_gbc = metrics.precision_score(y_test,y_test_gbc)
print("Gradient Boosting Classifier : precision on training Data: {:.3f}".format(precision_score_train_gbc))
print("Gradient Boosting Classifier : precision on test Data: {:.3f}".format(precision_score_test_gbc))
```

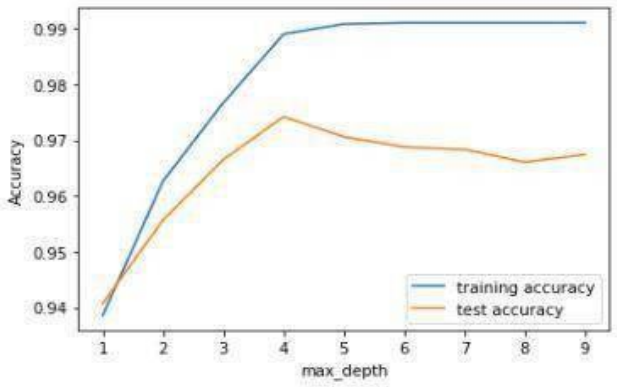
```
Gradient Boosting Classifier : Accuracy on training Data: 0.989
Gradient Boosting Classifier : Accuracy on test Data: 0.974

Gradient Boosting Classifier : f1_score on training Data: 0.990
Gradient Boosting Classifier : f1_score on test Data: 0.977

Gradient Boosting Classifier : Recall on training Data: 0.994
Gradient Boosting Classifier : Recall on test Data: 0.989
```

```
              precision    recall  f1-score   support

          -1       0.99      0.96      0.97       976
           1       0.97      0.99      0.98      1235

    accuracy                           0.97      2211
   macro avg       0.98      0.97      0.97      2211
weighted avg       0.97      0.97      0.97      2211
```

In [53]:
```
training_accuracy = []
test_accuracy = []
# try learning_rate from 0.1 to 0.9
depth = range(1,10)
for n in depth:
    forest_test =  GradientBoostingClassifier(learning_rate = n*0.1)

    forest_test.fit(X_train, y_train)
    # record training set accuracy
    training_accuracy.append(forest_test.score(X_train, y_train))
    # record generalization accuracy
    test_accuracy.append(forest_test.score(X_test, y_test))


#plotting the training & testing accuracy for n_estimators from 1 to 50
plt.figure(figsize=None)
plt.plot(depth, training_accuracy, label="training accuracy")
plt.plot(depth, test_accuracy, label="test accuracy")
plt.ylabel("Accuracy")
plt.xlabel("learning_rate")
plt.legend();
```



In [54]:
```
training_accuracy = []
test_accuracy = []
# try learning_rate from 0.1 to 0.9
depth = range(1,10,1)
for n in depth:
    forest_test =  GradientBoostingClassifier(max_depth=n,learning_rate = 0.7)

    forest_test.fit(X_train, y_train)
    # record training set accuracy
    training_accuracy.append(forest_test.score(X_train, y_train))
    # record generalization accuracy
    test_accuracy.append(forest_test.score(X_test, y_test))


#plotting the training & testing accuracy for n_estimators from 1 to 50
plt.figure(figsize=None)
plt.plot(depth, training_accuracy, label="training accuracy")
plt.plot(depth, test_accuracy, label="test accuracy")
plt.ylabel("Accuracy")
plt.xlabel("max_depth")
plt.legend();
```



In [55]:
```
#storing the results. The below mentioned order of parameter passing is important.

storeResults('Gradient Boosting Classifier',acc_test_gbc,f1_score_test_gbc,
             recall_score_train_gbc,precision_score_train_gbc)
```

Catboost

CatBoost is a recently open-sourced machine learning algorithm from Yandex. It can work with diverse data types to help solve a wide range of problems that businesses face today.

In [56]:
```
#  catboost Classifier Model
from catboost import CatBoostClassifier

# instantiate the model
cat = CatBoostClassifier(learning_rate  = 0.1)

# fit the model
cat.fit(X_train,y_train)
```

```
0:      learn: 0.5487232        total: 90.5ms   remaining: 1m 30s
1:      learn: 0.4349357        total: 97.8ms   remaining: 48.8s
2:      learn: 0.3609236        total: 104ms    remaining: 34.7s
3:      learn: 0.3050829        total: 111ms    remaining: 27.6s
4:      learn: 0.2766620        total: 117ms    remaining: 23.3s
5:      learn: 0.2475476        total: 124ms    remaining: 20.5s
6:      learn: 0.2286637        total: 129ms    remaining: 18.3s
7:      learn: 0.2138754        total: 133ms    remaining: 16.5s
8:      learn: 0.2013643        total: 138ms    remaining: 15.2s
9:      learn: 0.1896378        total: 143ms    remaining: 14.1s
10:     learn: 0.1819539        total: 148ms    remaining: 13.3s
11:     learn: 0.1767867        total: 153ms    remaining: 12.6s
12:     learn: 0.1727735        total: 158ms    remaining: 12s
13:     learn: 0.1682578        total: 164ms    remaining: 11.5s
14:     learn: 0.1641759        total: 169ms    remaining: 11.1s
15:     learn: 0.1614218        total: 174ms    remaining: 10.7s
16:     learn: 0.1558968        total: 179ms    remaining: 10.4s
17:     learn: 0.1535881        total: 184ms    remaining: 10.1s
18:     learn: 0.1514228        total: 189ms    remaining: 9.77s
```

In [57]:
```
#predicting the target value from the model for the samples
y_train_cat = cat.predict(X_train)
y_test_cat = cat.predict(X_test)
```

In [58]:
```
#computing the accuracy, f1_score, Recall, precision of the model performance

acc_train_cat  = metrics.accuracy_score(y_train,y_train_cat)
acc_test_cat = metrics.accuracy_score(y_test,y_test_cat)
print("CatBoost Classifier : Accuracy on training Data: {:.3f}".format(acc_train_cat))
print("CatBoost Classifier : Accuracy on test Data: {:.3f}".format(acc_test_cat))
print()

f1_score_train_cat = metrics.f1_score(y_train,y_train_cat)
f1_score_test_cat = metrics.f1_score(y_test,y_test_cat)
print("CatBoost Classifier : f1_score on training Data: {:.3f}".format(f1_score_train_cat))
print("CatBoost Classifier : f1_score on test Data: {:.3f}".format(f1_score_test_cat))
print()
```

```
precision_score_train_cat = metrics.precision_score(y_train,y_train_cat)
precision_score_test_cat = metrics.precision_score(y_test,y_test_cat)
print("CatBoost Classifier : precision on training Data: {:.3f}".format(precision_score_train_cat))
print("CatBoost Classifier : precision on test Data: {:.3f}".format(precision_score_test_cat))
```

```
CatBoost Classifier : Accuracy on training Data: 0.991
CatBoost Classifier : Accuracy on test Data: 0.972

CatBoost Classifier : f1_score on training Data: 0.992
CatBoost Classifier : f1_score on test Data: 0.975

CatBoost Classifier : Recall on training Data: 0.994
CatBoost Classifier : Recall on test Data: 0.982

CatBoost Classifier : precision on training Data: 0.989
CatBoost Classifier : precision on test Data: 0.969
```

In [59]:
```
#computing the classification report of the model

print(metrics.classification_report(y_test, y_test_cat))
```

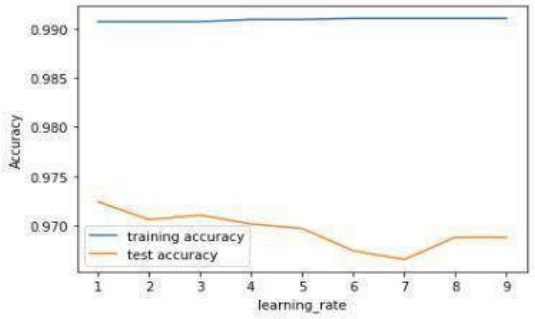
```
              precision    recall  f1-score   support

          -1       0.98      0.96      0.97       976
           1       0.97      0.98      0.98      1235

    accuracy                           0.97      2211
   macro avg       0.97      0.97      0.97      2211
weighted avg       0.97      0.97      0.97      2211
```

In [60]:
```
training_accuracy = []
test_accuracy = []
# try learning_rate from 0.1 to 0.9
depth = range(1,10)
for n in depth:
    forest_test =  CatBoostClassifier(learning_rate = n*0.1)

    forest_test.fit(X_train, y_train)
    # record training set accuracy
    training_accuracy.append(forest_test.score(X_train, y_train))
    # record generalization accuracy
    test_accuracy.append(forest_test.score(X_test, y_test))
```

```
1:      learn: 0.4349357        total: 10.1ms   remaining: 5.06s
2:      learn: 0.3609236        total: 16.6ms   remaining: 5.53s
3:      learn: 0.3050829        total: 22.9ms   remaining: 5.7s
4:      learn: 0.2766620        total: 29.2ms   remaining: 5.81s
5:      learn: 0.2475476        total: 35.5ms   remaining: 5.88s
6:      learn: 0.2286637        total: 46.8ms   remaining: 6.63s
7:      learn: 0.2138754        total: 53.2ms   remaining: 6.6s
8:      learn: 0.2013643        total: 59.7ms   remaining: 6.57s
9:      learn: 0.1896378        total: 67.3ms   remaining: 6.66s
10:     learn: 0.1819539        total: 75.6ms   remaining: 6.8s
11:     learn: 0.1767867        total: 81.3ms   remaining: 6.7s
12:     learn: 0.1727735        total: 87ms     remaining: 6.61s
13:     learn: 0.1682578        total: 93.9ms   remaining: 6.61s
14:     learn: 0.1641759        total: 100ms    remaining: 6.59s
15:     learn: 0.1614218        total: 107ms    remaining: 6.56s
16:     learn: 0.1558968        total: 113ms    remaining: 6.51s
17:     learn: 0.1535881        total: 119ms    remaining: 6.48s
18:     learn: 0.1514228        total: 125ms    remaining: 6.45s
```

In [61]:

```
#plotting the training & testing accuracy for n_estimators from 1 to 50
plt.figure(figsize=None)
plt.plot(depth, training_accuracy, label="training accuracy")
plt.plot(depth, test_accuracy, label="test accuracy")
plt.ylabel("Accuracy")
plt.xlabel("learning_rate")
plt.legend();
```



In [62]:

```
#storing the results. The below mentioned order of parameter passing is important.

storeResults('CatBoost Classifier',acc_test_cat,f1_score_test_cat,
             recall_score_train_cat,precision_score_train_cat)
```

Xgboost

XGBoost is an implementation of gradient boosted decision trees designed for speed and performance that is dominative competitive machine learning.

In [65]:
```
#computing the accuracy, f1_score, Recall, precision of the model performance

acc_train_xgb = metrics.accuracy_score(y_train,y_train_xgb)
acc_test_xgb = metrics.accuracy_score(y_test,y_test_xgb)
print("XGBoost Classifier : Accuracy on training Data: {:.3f}".format(acc_train_xgb))
print("XGBoost Classifier : Accuracy on test Data: {:.3f}".format(acc_test_xgb))
print()

f1_score_train_xgb = metrics.f1_score(y_train,y_train_xgb)
f1_score_test_xgb = metrics.f1_score(y_test,y_test_xgb)
print("XGBoost Classifier : f1_score on training Data: {:.3f}".format(f1_score_train_xgb))
print("XGBoost Classifier : f1_score on test Data: {:.3f}".format(f1_score_test_xgb))
print()

recall_score_train_xgb = metrics.recall_score(y_train,y_train_xgb)
recall_score_test_xgb = metrics.recall_score(y_test,y_test_xgb)
print("XGBoost Classifier : Recall on training Data: {:.3f}".format(recall_score_train_xgb))
print("XGBoost Classifier : Recall on test Data: {:.3f}".format(recall_score_train_xgb))
print()

precision_score_train_xgb = metrics.precision_score(y_train,y_train_xgb)
precision_score_test_xgb = metrics.precision_score(y_test,y_test_xgb)
print("XGBoost Classifier : precision on training Data: {:.3f}".format(precision_score_train_xgb))
print("XGBoost Classifier : precision on test Data: {:.3f}".format(precision_score_train_xgb))
```

```
XGBoost Classifier : Accuracy on training Data: 0.987
XGBoost Classifier : Accuracy on test Data: 0.969

XGBoost Classifier : f1_score on training Data: 0.988
XGBoost Classifier : f1_score on test Data: 0.973

XGBoost Classifier : Recall on training Data: 0.993
XGBoost Classifier : Recall on test Data: 0.993

XGBoost Classifier : precision on training Data: 0.984
XGBoost Classifier : precision on test Data: 0.984
```

In [66]:
```
#storing the results. The below mentioned order of parameter passing is important.

storeResults('XGBoost Classifier',acc_test_xgb,f1_score_test_xgb,
             recall_score_train_xgb,precision_score_train_xgb)
```

Multilayer Perceptron

Multilayer perceptron Classifier stands for Multi-layer Perceptron classifier which in the name itself connects to a Neural Network. It is a feed forward artificial neural network. The output of the first layer is the input of the second layer and output of the second layer is the input of third layer.

In [67]:
```
# Multi-layer Perceptron Classifier Model
from sklearn.neural_network import MLPClassifier

# instantiate the model
mlp = MLPClassifier()
#mlp = GridSearchCV(mlpc, parameter_space)

# fit the model
mlp.fit(X_train,y_train)
```

Out[67]:
```
MLPClassifier()
```

In [68]:
```
#predicting the target value from the model for the samples
y_train_mlp = mlp.predict(X_train)
y_test_mlp = mlp.predict(X_test)
```

In [69]:
```
#computing the accuracy, f1_score, Recall, precision of the model performance

acc_train_mlp  = metrics.accuracy_score(y_train,y_train_mlp)
acc_test_mlp = metrics.accuracy_score(y_test,y_test_mlp)
print("Multi-layer Perceptron : Accuracy on training Data: {:.3f}".format(acc_train_mlp))
print("Multi-layer Perceptron : Accuracy on test Data: {:.3f}".format(acc_test_mlp))
print()

f1_score_train_mlp = metrics.f1_score(y_train,y_train_mlp)
f1_score_test_mlp = metrics.f1_score(y_test,y_test_mlp)
print("Multi-layer Perceptron : f1_score on training Data: {:.3f}".format(f1_score_train_mlp))
print("Multi-layer Perceptron : f1_score on test Data: {:.3f}".format(f1_score_train_mlp))
print()

recall_score_train_mlp = metrics.recall_score(y_train,y_train_mlp)
recall_score_test_mlp = metrics.recall_score(y_test,y_test_mlp)
print("Multi-layer Perceptron : Recall on training Data: {:.3f}".format(recall_score_train_mlp))
print("Multi-layer Perceptron : Recall on test Data: {:.3f}".format(recall_score_test_mlp))
print()

precision_score_train_mlp = metrics.precision_score(y_train,y_train_mlp)
precision_score_test_mlp = metrics.precision_score(y_test,y_test_mlp)
print("Multi-layer Perceptron : precision on training Data: {:.3f}".format(precision_score_train_mlp))
print("Multi-layer Perceptron : precision on test Data: {:.3f}".format(precision_score_test_mlp))
```

```
Multilayer Perceptron : Accuracy on training Data: 0.986
Multilayer Perceptron : Accuracy on test Data: 0.969

Multilayer Perceptron : f1_score on training Data: 0.988
Multilayer Perceptron : f1_score on test Data: 0.988
```

4.3 Model Evaluation

The models are evaluated and considered metric is accuracy.

Below figure shows the training and test dataset accuracy by the respective models.

SL. No	Machine Learning Model	Train Accuracy	Test Accuracy
1	Logistic Regression	0.927	0.934
2	K-nearest Neighbour	0.989	0.956
3	Support Vector classifier	0.969	0.964
4	Naïve Bayes	0.605	0.605
5	Decision tree	0.991	0.960
6	Gradient Boasting Classifier	0.989	0.974
7	Random Forest	0.991	0.967
8	Catboast	0.991	0.972
9	XGBoast Classifier	0.987	0.969
10	Multilayer Perceptron	0.986	0.969

Table 1. Model Evaluation

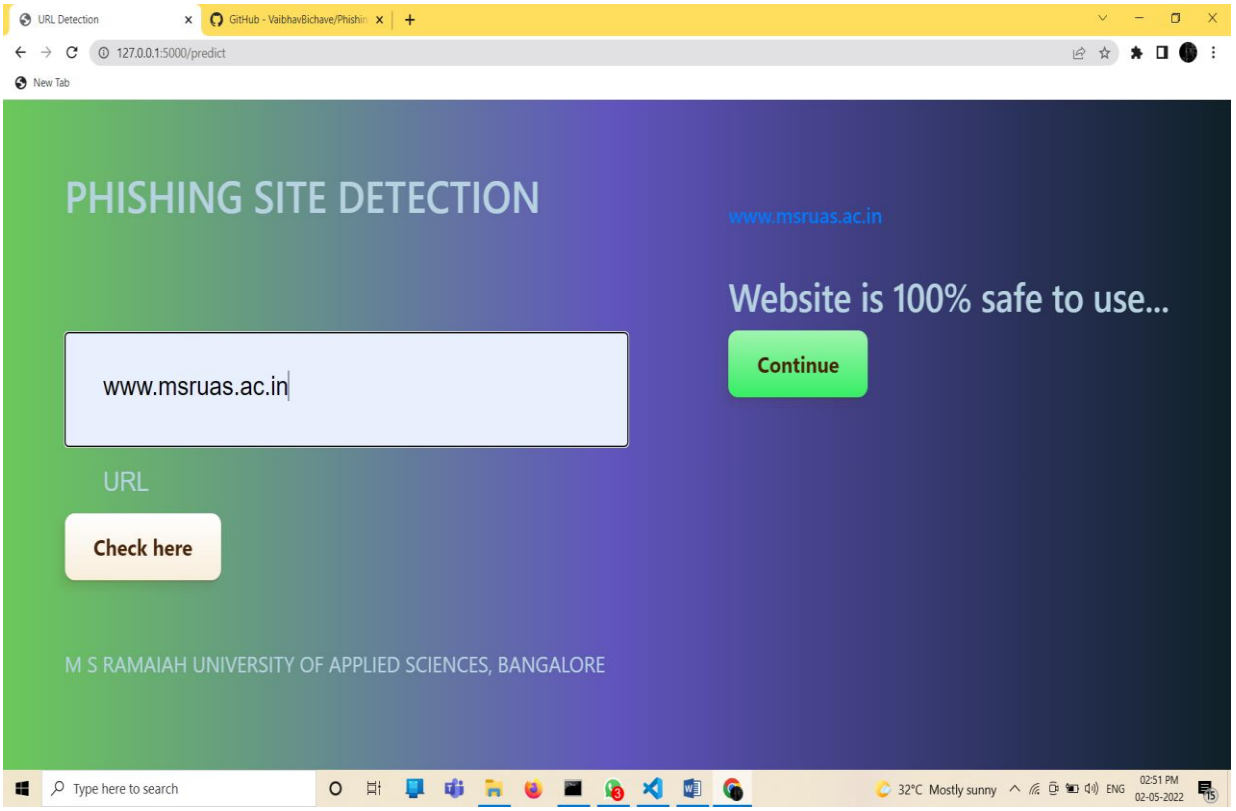
For the above it is clear that the Gradient Boasting Classifier model gives the better performance.

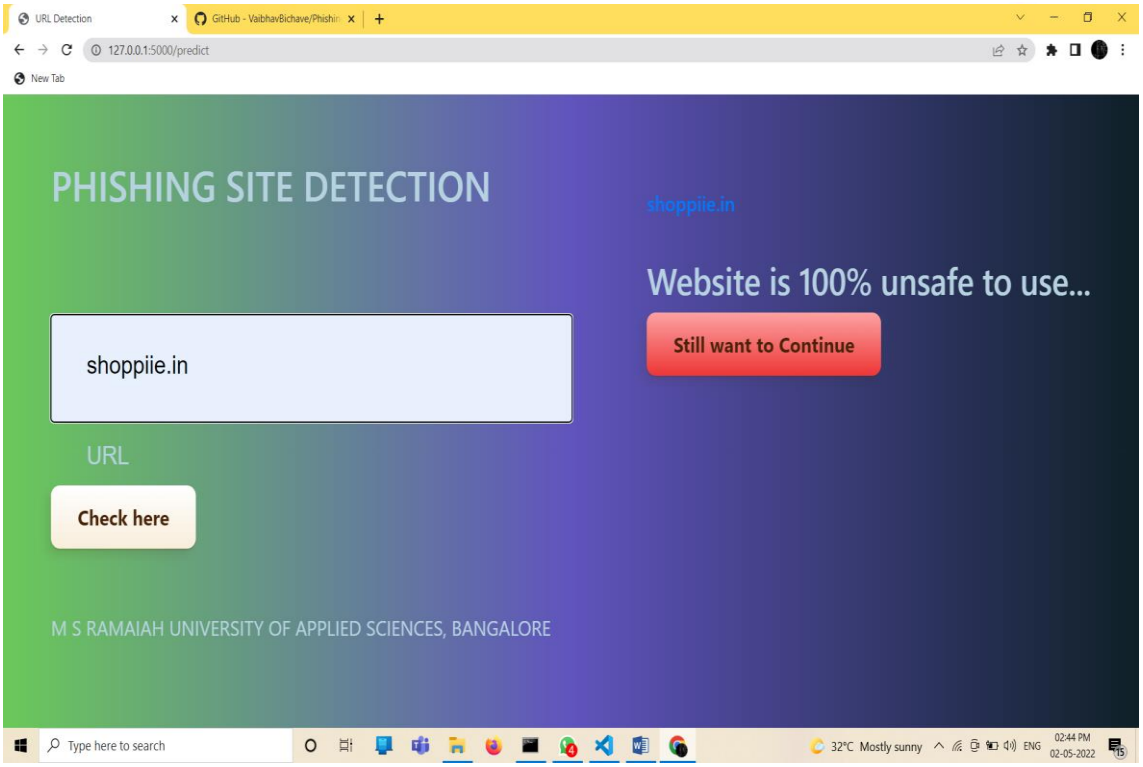
We achieved 97.4% detection accuracy using Gradient Boasting Classifier algorithm with lowest false positive rate and hence reduces the chance of malicious attachments. Also result shows that classifiers give better performance when we used more data as training data.

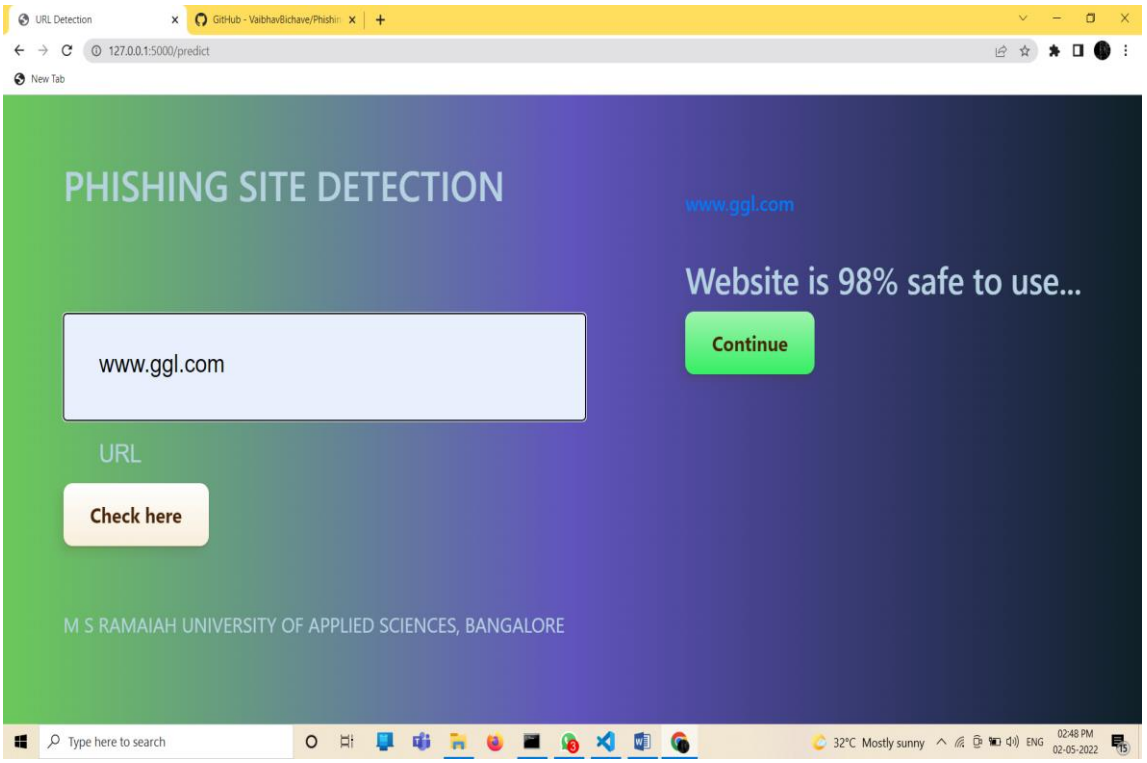
5. Results

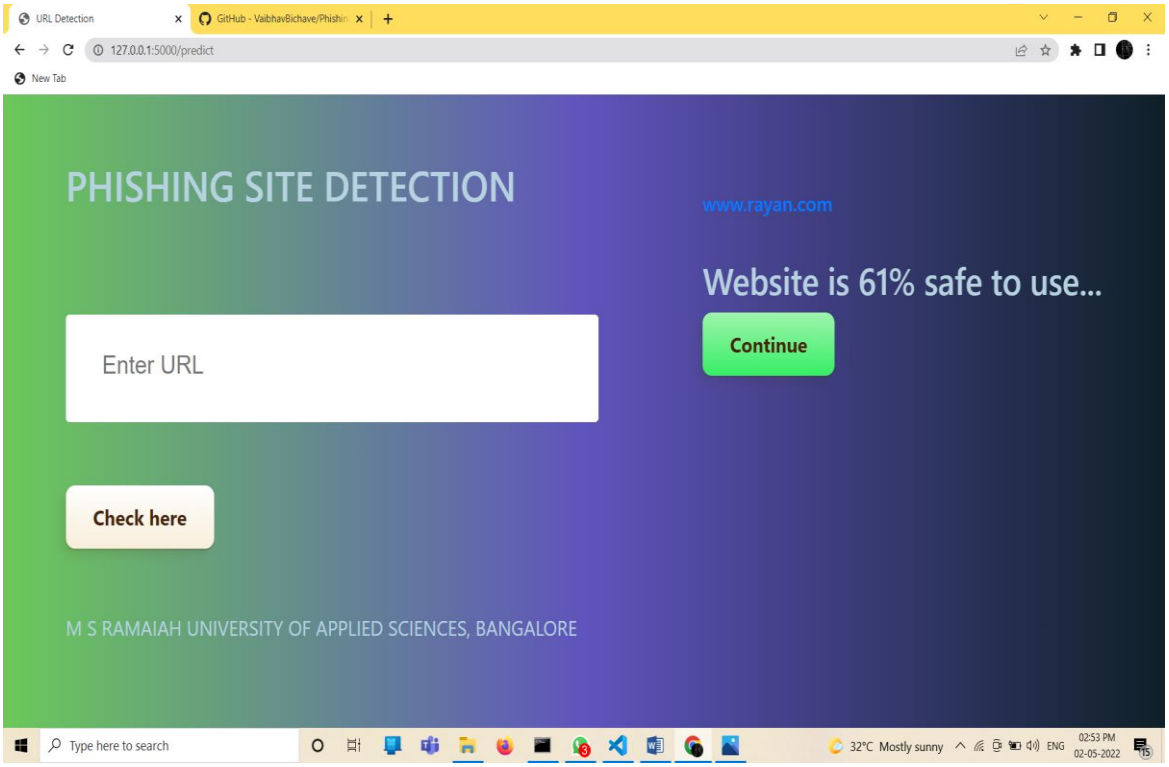
The Phishing site detection application for demonstrating the use of Machine Learning Was developed successfully. Used python libraries to execute the code and different Machine learning algorithms.

5.1 Result




URL Detection
GitHub - VaibhavBichave/Phishin
127.0.0.1:5000/predict
New Tab
PHISHING SITE DETECTION
shoppiie.in
Website is 100% unsafe to use...
Still want to Continue
shoppiie.in
URL
Check here
M S RAMAIAH UNIVERSITY OF APPLIED SCIENCES, BANGALORE
Type here to search
32°C Mostly sunny
ENG
02:44 PM
02-05-2022


URL Detection
GitHub - VaibhavBichave/Phishin
127.0.0.1:5000/predict
New Tab
PHISHING SITE DETECTION
www.ggl.com
Website is 98% safe to use...
Continue
www.ggl.com
URL
Check here
M S RAMAIAH UNIVERSITY OF APPLIED SCIENCES, BANGALORE
Type here to search
32°C Mostly sunny
ENG
02:48 PM
02-05-2022



5.2 Summary

We observed two perspectives from the existing detection schemes

(i) dataset perspective

(ii) Feature perspective.

From the dataset perspective, researchers primarily analyze the detection method on imbalanced data, in which the majority class is legitimate sites. This results in a Biasing majority class. Put differently, the result is biased although it has a high false positive rate. To address this, oversampling on minority data becomes effective since it balances data size by realistic automated minority-class data. From the feature perspective, we find that several URL based features–such as the number of subdomains and URL length could also be biased since they highly rely on the dataset. In other words, many researchers use Alexa.com for legitimate dataset, in which only index pages of highly ranked websites are provided. However, phishing datasets from PhishTank.com or OpenPhish.com list the entire URLs of the phishing webpages in which phishers use free hosting services that are highly ranked in Alexa. Thus, as for the number of subdomains, legitimate sites from Alexa.com will not have any, while phishing sites will. Furthermore, phishers have complete control over URL composition except for the domain name. Features like URL length can be easily manipulated. Therefore, researchers have recently targeted domain name-based features instead of entire URL to extract characteristics of domain name and current page content.

Project Cost Estimation

The cost of project is summarized in a tabular form displayed below:

Table 3 Cost estimation table

Serial Number	Resources and Work	Cost(Rs)
1	Laptop (for development)	30,000/-
2	Human Resources (5 * 5,000)	25,000/-
	TOTAL	**55,000/-**

6.1 Conclusion

This project started with introduction. Background Theory of all resources including technology used, Framework selected, IDE's worked upon and engines applied were extensively elaborated so that this theory can be applied effectively and the reason for their use application can be well understood. Later, all objectives were listed done after declaring title and Aim of the project and methods and mythologies in order to complete the objectives were well tabulated. Later, block diagram was created giving complete view of the project from different perspective. Implementation of the project was displayed via displaying code written in JavaScript programming language interacting with each other

to create the web application. Testing was done for all functionalities and were found to be working successfully. Later in result section, all screenshot of application in different states were taken in order to demonstrate the end product of this project. Each screenshot was explained with its importance as a view for the web application. tis found that phishing attacks is very crucial and it is important for us to get a mechanism to detect it, As very important and personal information of the user can be leaked through phishing websites, it becomes more critical to take care of this issue. This problem can be easily solved by using any of the machine learning algorithm with the classifier.

We already have classifiers which gives good prediction rate of the phishing beside, but after our survey that it will be better to use a hybrid approach for the prediction and further improve the accuracy prediction rate of phishing websites. We have seen that existing system gives less accuracy so we proposed a new phishing method that employs URL based features and also we generated classifiers through several machine learning.

Phishing is a way to obtain users private information via email or website. As technology increases, phishing attackers using new methods day by day. In this report describe detailed literature survey about phishing website detection. Phishing websites are short-lived, and thousands of fake websites are generated every day. Therefore, there is requirement of real- time, fast and intelligent phishing detection solution. According to this, Machine learning is efficient technique to detect phishing. More features can be added to improve the accuracy of the proposed phishing detection system.

6.2 SUGGESTION FOR FUTURE WORK

Future scope

In future in the event that we get structured dataset of phishing we can perform phishing recognition substantially more quicker than some other technique.In future we can utilize a mix of some other at least two classifier to get greatest precision. We likewise plan to investigate different phishing methods that utilizes Lexical elements, Network based features,Content based highlights, Webpage based highlights and HTML and JavaScript elements of pages which can work on the presentation of the system. Specifically, we remove highlights from URLs and pass it through the different classifiers.

References

1. The dataset is borrowed from Kaggle, https://www.kaggle.com/eswarchandt/phishing-website-detector
2. Kulkarni, Arun D. and Brown,, Leonard L. III, "Phishing Websites Detection using Machine Learning" (2019). *Computer Science Faculty Publications and Presentations.* Paper 20. http://hdl.handle.net/10950/1862
3. Hossein Shirazi, Kyle Haefner, Indrakshi Ray: Fresh-Phish: A Framework for Auto-Detection of Phishing Websites: In (International Conference on Information Reuse and Integration (IRI)) IEEE,2017.
4. Ankit Kumar Jain, B. B. Gupta : Towards detection of phishing websites on client-side using machine learning based approach :In Springer Science+Business Media, LLC, part of Springer Nature 2017
5. Bhagyashree E. Sananse, Tanuja K. Sarode : Phishing URL Detection: A Machine Learning and Web Mining-based Approach : In International Journal of Computer Applications,2015
6. The dataset is borrowed from Kaggle, https://www.kaggle.com/eswarchandt/phishing-website-detector .