

REAL AND FAKE JOB POSTING USING MACHINE LEARNING TECHNIQUE

Submitted in partial fulfillment of the requirements

for the award of

Bachelor of Engineering degree in Computer Science and Engineering

By

V. Arthi Nandhini

(38110049)

M. Keerthigha

(38110248)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

**SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade “A” by NAAC**

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

MARCH - 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **ARTHI NANDHINI V. (38110049)** **KEERTHIGHA M. (38110248)** who carried out the project entitled **"REAL AND FAKE JOB POSTING USING MACHINE LEARNING TECHNIQUE"** under my supervision from November 2021 to March 2022.

INTERNAL GUIDE

Dr. J. Refonaa

Head of the Department

Dr.A.Lakshmanan,ME,Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

We M.KEERTHIGHA and V.ARTHI NANDHINI hereby declare that the Project Report entitled REAL AND FAKE JOB POSTING USING MACHINE LEARNING TECHNIQUE done under the guidance of Dr. J. Refonaa(Internal) is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering / Technology degree in Computer science and engineering.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATES:

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D, Dean**, School of Computing

Dr. L. Lakshmanan M.E., Ph.D. , and Dr.S.Vigneshwari M.E., Ph.D. Heads of the Department of Computer Science and Engineering for providing necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide Dr./Mr./Ms for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT:

Everything is being online now it allows people to reduce their manual efforts all of the job postings are posted online now so it gives the company a wide range of area to gather the talented candidates and for the people who are searching they can also know the information most companies can directly post the job. All job postings which are posted are not true there are fraudulent job postings also. So we try to classify the fraudulent posting from real. The aim is to predict machine learning based techniques for real or fake job prediction results in best accuracy. The analysis of dataset is done by supervised machine learning technique(SMLT) to capture several information's like, variable identification, uni-variate analysis, bi-variate and multi-variate analysis, missing value treatments and analyze the data validation, data cleaning/preparing and data visualization will be done on the entire given dataset.

TABLE OF CONTENT

SL.NO	TITLE	PAGE.NO
01	ABSTRACT	4
02	CHAPTER 1 : INTRODUCTION DRAWBACKS	11
03	DATA SCIENCE ARTIFICIAL INTELLIGENCE	12 14
04	MACHINE LEARNING	18
05	PREPARING DATASET	21
06	PROPOSED SYSTEM ADVANTAGES	21 23
07	CHAPTER 2 : LITERATURE SURVEY	24
08	CHAPTER 3 : METHODOLOGY OBJECTIVES PROJECT GOAL SCOPE OF THE PROJECT	29 29 29 30
09	FEASIBILITY STUDY	30
10	LIST OF MODULES	32
11	PROJECT REQUIREMENTS FUNCTIONAL REQUIREMENTS NON-FUNCTIONAL REQUIREMENTS	33 33 33
12	ENVIRONMENT REQUIREMENT	34
13	SOFTWARE DESCRIPTION ANACONDA NAVIGATOR JUPYTER NOTEBOOK	35 36 40
14	PYTHON	46
15	SYSTEM ARCHITECTURE	60
16	WORKFLOW DIAGRAM	61
17	USECASE DIAGRAM	62
18	CLASS DIAGRAM	63
19	ACTIVITY DIAGRAM	64
20	SEQUENCE DIAGRAM	65
21	ER – DIAGRAM	66
22	MODULE DESCRIPTION	71

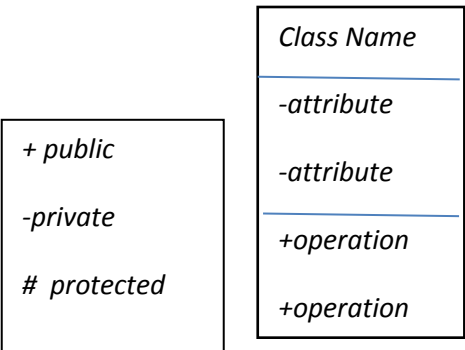
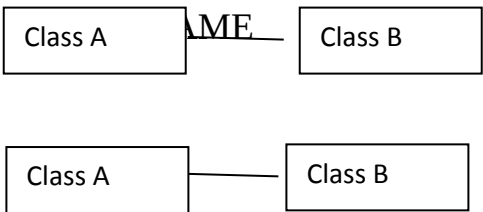
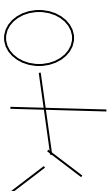
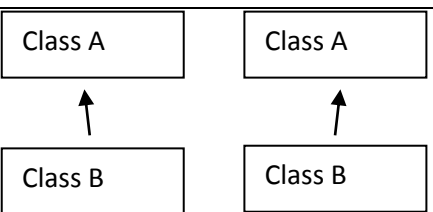
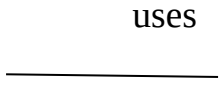
	MODULE DIAGRAM	71
	MODULE GIVEN INPUT EXPECTED OUTPUT	74
23	CHAPTER 4 : RESULT AND DISCUSSION	67
24	CHAPTER 5 : CONCLUSION AND FUTURE WORK	87
25	APPENDICES	88
	A.SAMPLE CODE	88
	B.SCREENSHOTS	105
	C.PAPER PUBLICATION WITH PLAGIARISM REPORT	107

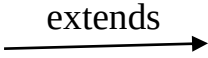
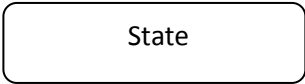
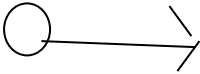
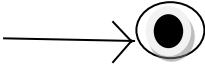
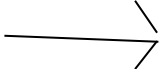
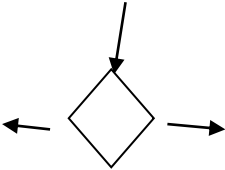
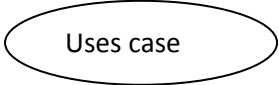
LIST OF FIGURES

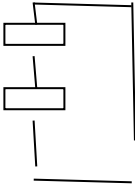
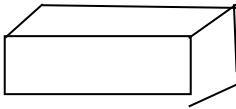
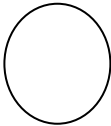
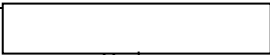
SL.NO	TITLE	PAGE.NO
01	SYSTEM ARCHITECTURE	60
02	WORKFLOW DIAGRAM	61
03	USECASE DIAGRAM	62
04	CLASS DIAGRAM	63
05	ACTIVITY DIAGRAM	64
06	SEQUENCE DIAGRAM	65
07	ER – DIAGRAM	66
08	MODULE DIAGRAM	71

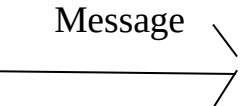
LIST OF SYMBOLS

S.NO	NOTATION NAME	NOTATION	DESCRIPTION

1.	Class		Represents a collection of similar entities grouped together.
2.	Association		Associations represents static relationships between classes. Roles represents the way the two classes see each other.
3.	Actor		It aggregates several classes into a single classes.
4.	Aggregation		Interaction between the system and external environment
5.	Relation(uses)		Used for additional process communication.
6.	Relation		Extends

	(extends)		relationship is used when one use case is similar to another use case but does a bit more.
7.	Communication		Communication between various use cases.
8.	State		State of the process.
9.	Initial State		Initial state of the object
10.	Final state		Final state of the object
11.	Control flow		Represents various control flow between the states.
12.	Decision box		Represents decision making process from a constraint
13.	Use case		Interaction between the system

			and external environment.
14.	Component		Represents physical modules which is a collection of components.
15.	Node		Represents physical modules which are a collection of components
16.	Data Process/State		A circle in DFD represents a state or process which has been triggered due to some event or action.
17.	External entity		Represents external entities such as keyboard, sensors etc.
18.	Transition		Represents communication that occurs between processes.
19.	Object Lifeline		Represents the

			vertical dimensions that the object communications.
20.	Message		Represents the message exchanged.

CHAPTER 1 : INTRODUCTION

The existing data-driven approaches typically capture credibility-indicative representations from relevant articles for fake news detection, such as skeptical and conflicting opinions. However, these methods still have several drawbacks: Due to the difficulty of collecting fake news, the capacity of the existing datasets is relatively small; and there is considerable unverified news that lacks conflicting voices in relevant articles, which makes it difficult for the existing methods to identify their credibility. Especially, the differences between true and fake news are not limited to whether there are conflict features in their relevant articles, but also include more extensive hidden differences at the linguistic level, such as the perspectives of emotional expression (like extreme emotion in fake news), writing style (like the shocking title in clickbait), etc., the existing methods are difficult to fully capture these differences.

Drawbacks:

- They are not using machine learning
- Accuracy , Recall F1 score metrics are not calculated and machine learning algorithms are not applied

Data Science

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from structured and unstructured data, and apply knowledge and actionable insights from data across a broad range of application domains.

The term "data science" has been traced back to 1974, when Peter Naur proposed it as an alternative name for computer science. In 1996, the International Federation of Classification Societies became the first conference to

specifically feature data science as a topic. However, the definition was still in flux.

The term “data science” was first coined in 2008 by D.J. Patil, and Jeff Hammerbacher, the pioneer leads of data and analytics efforts at LinkedIn and Facebook. In less than a decade, it has become one of the hottest and most trending professions in the market.

Data science is the field of study that combines domain expertise, programming skills, and knowledge of mathematics and statistics to extract meaningful insights from data.

Data science can be defined as a blend of mathematics, business acumen, tools, algorithms and machine learning techniques, all of which help us in finding out the hidden insights or patterns from raw data which can be of major use in the formation of big business decisions.

Data Scientist:

Data scientists examine which questions need answering and where to find the related data. They have business acumen and analytical skills as well as the ability to mine, clean, and present data. Businesses use data scientists to source, manage, and analyze large amounts of unstructured data.

Required Skills for a Data Scientist:

- **Programming:** Python, SQL, Scala, Java, R, MATLAB.

- **Machine Learning:** Natural Language Processing, Classification, Clustering.
- **Data Visualization:** Tableau, SAS, D3.js, Python, Java, R libraries.
- **Big data platforms:** MongoDB, Oracle, Microsoft Azure, Cloudera.

ARTIFICIAL INTELLIGENCE

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving.

Artificial intelligence (AI) is intelligence demonstrated by machines, as opposed to the natural intelligence displayed by humans or animals. Leading AI textbooks define the field as the study of "intelligent agents" any system that perceives its environment and takes actions that maximize its chance of achieving its goals. Some popular accounts use the term "artificial intelligence" to describe machines that mimic "cognitive" functions that humans associate with the human mind, such as "learning" and "problem solving", however this definition is rejected by major AI researchers.

Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. Specific applications of AI include expert systems, natural language processing, speech recognition and machine vision.

AI applications include advanced web search engines, recommendation systems (used by Youtube, Amazon and Netflix), Understanding human speech (such as Siri or Alexa), self-driving cars (e.g. Tesla), and competing at the highest level in strategic game systems (such as chess and Go), As machines become increasingly capable, tasks considered to require "intelligence" are often

removed from the definition of AI, a phenomenon known as the AI effect. For instance, optical character recognition is frequently excluded from things considered to be AI, having become a routine technology.

Artificial intelligence was founded as an academic discipline in 1956, and in the years since has experienced several waves of optimism, followed by disappointment and the loss of funding (known as an "AI winter"), followed by new approaches, success and renewed funding. AI research has tried and discarded many different approaches during its lifetime, including simulating the brain, modeling human problem solving, formal logic, large databases of knowledge and imitating animal behavior. In the first decades of the 21st century, highly mathematical statistical machine learning has dominated the field, and this technique has proved highly successful, helping to solve many challenging problems throughout industry and academia.

The various sub-fields of AI research are centered around particular goals and the use of particular tools. The traditional goals of AI research include reasoning, knowledge representation, planning, learning, natural language processing, perception and the ability to move and manipulate objects. General intelligence (the ability to solve an arbitrary problem) is among the field's long-term goals. To solve these problems, AI researchers use versions of search and mathematical optimization, formal logic, artificial neural networks, and methods based on statistics, probability and economics. AI also draws upon computer science, psychology, linguistics, philosophy, and many other fields.

The field was founded on the assumption that human intelligence "can be so precisely described that a machine can be made to simulate it". This raises philosophical arguments about the mind and the ethics of creating artificial beings endowed with human-like intelligence. These issues have been explored

by myth, fiction and philosophy since antiquity. Science and futurology have also suggested that, with its enormous potential and power, AI may become an existential risk to humanity.

As the hype around AI has accelerated, vendors have been scrambling to promote how their products and services use AI. Often what they refer to as AI is simply one component of AI, such as machine learning. AI requires a foundation of specialized hardware and software for writing and training machine learning algorithms. No one programming language is synonymous with AI, but a few, including Python, R and Java, are popular.

In general, AI systems work by ingesting large amounts of labeled training data, analyzing the data for correlations and patterns, and using these patterns to make predictions about future states. In this way, a chatbot that is fed examples of text chats can learn to produce life like exchanges with people, or an image recognition tool can learn to identify and describe objects in images by reviewing millions of examples.

AI programming focuses on three cognitive skills: learning, reasoning and self-correction.

Learning processes. This aspect of AI programming focuses on acquiring data and creating rules for how to turn the data into actionable information. The rules, which are called algorithms, provide computing devices with step-by-step instructions for how to complete a specific task.

Reasoning processes. This aspect of AI programming focuses on choosing the right algorithm to reach a desired outcome.

Self-correction processes. This aspect of AI programming is designed to continually fine-tune algorithms and ensure they provide the most accurate results possible.

AI is important because it can give enterprises insights into their operations that they may not have been aware of previously and because, in some cases, AI can perform tasks better than humans. Particularly when it comes to repetitive, detail-oriented tasks like analyzing large numbers of legal documents to ensure relevant fields are filled in properly, AI tools often complete jobs quickly and with relatively few errors.

Artificial neural networks and deep learning artificial intelligence technologies are quickly evolving, primarily because AI processes large amounts of data much faster and makes predictions more accurately than humanly possible.

Natural Language Processing (NLP):

Natural language processing (NLP) allows machines to read and understand human language. A sufficiently powerful natural language processing system would enable natural-language user interfaces and the acquisition of knowledge directly from human-written sources, such as newswire texts. Some straightforward applications of natural language processing include information retrieval, text mining, question answering and machine translation. Many current approaches use word co-occurrence frequencies to construct syntactic representations of text.

"Keyword spotting" strategies for search are popular and scalable but dumb; a search query for "dog" might only match documents with the literal word "dog" and miss a document with the word "poodle". "Lexical affinity" strategies use the occurrence of words such as "accident" to assess the sentiment of a document. Modern statistical NLP approaches can combine all these strategies as well as others, and often achieve acceptable accuracy at the page or paragraph level. Beyond semantic NLP, the ultimate goal of "narrative" NLP is to embody a full understanding of commonsense reasoning. By 2019, transformer-based deep learning architectures could generate coherent text.

MACHINE LEARNING

Machine learning is to predict the future from past data. Machine learning (ML) is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data and the basics of Machine Learning, implementation of a simple machine learning algorithm using python. Process of training and prediction involves use of specialized algorithms. It feed the training data to an algorithm, and the algorithm uses this training data to give predictions on a new test data. Machine learning can be roughly separated in to three categories. There are supervised learning,

unsupervised learning and reinforcement learning. Supervised learning program is both given the input data and the corresponding labeling to learn data has to be labeled by a human being beforehand. Unsupervised learning is no labels. It provided to the learning algorithm. This algorithm has to figure out the clustering of the input data. Finally, Reinforcement learning dynamically interacts with its environment and it receives positive or negative feedback to improve its performance.

Data scientists use many different kinds of machine learning algorithms to discover patterns in python that lead to actionable insights. At a high level, these different algorithms can be classified into two groups based on the way they “learn” about data to make predictions: supervised and unsupervised learning. Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modeling is the task of approximating a mapping function from input variables(X) to discrete output variables(y). In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc.



Supervised Machine Learning is the majority of practical machine learning uses supervised learning. Supervised learning is where have input variables (X) and an output variable (y) and use an algorithm to learn the mapping function from the input to the output is $y = f(X)$. The goal is to approximate the mapping function so well that when you have new input data (X) that you can predict the output variables (y) for that data. Techniques of Supervised Machine Learning algorithms include logistic regression, multi-class classification, Decision Trees and support vector machines etc. Supervised learning requires that the data used to train the algorithm is already labelled with correct answers. Supervised learning problems can be further grouped into Classification problems. This problem has as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables. The difference between the two tasks is the fact that the dependent attribute is numerical for categorical for classification. A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes. A classification problem is when the output variable is a category, such as “red” or “blue”.

PREPARING THE DATASET:

This dataset contains 1400 records of features, which were then classified into 2 classes:

- REAL
- FAKE

Proposed System:

The proposed method is built a machine learning model to classify the real or fake job posting to overcome this method to implement machine learning approach. The dataset is first preprocessed and the columns are analyzed to see the dependent and independent variable and then different machine learning algorithms would be applied to extract patterns and to obtain results with maximum accuracy.

Exploratory Data Analysis

Multiple datasets from different sources would be combined to form a generalized dataset, and then different machine learning algorithms would be applied to extract patterns and to obtain results with maximum accuracy.

Data Wrangling

In this section of the report will load in the data, check for cleanliness, and then trim and clean given dataset for analysis. Make sure that the document steps carefully and justify for cleaning decisions.

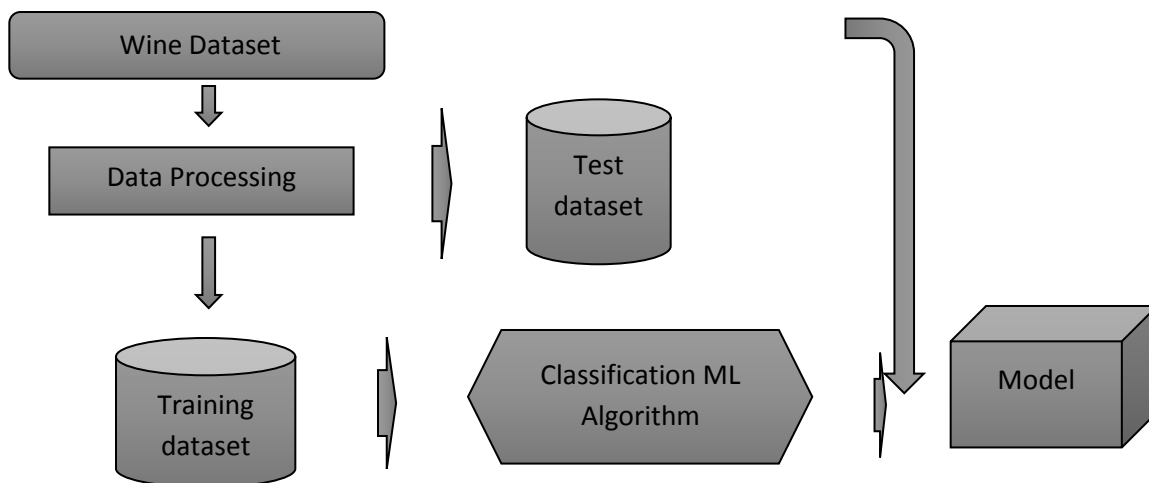
Data collection

The data set collected for predicting given data is split into Training set and Test set. Generally, 7:3 ratios are applied to split the Training set and Test set. The Data Model which was created using machine learning algorithms are applied on the Training set and based on the test result accuracy, Test set prediction is done.

Building the classification model

It provides better results in classification problem.

- It is strong in preprocessing outliers, irrelevant variables, and a mix of continuous, categorical and discrete variables.
- It produces out of bag estimate error which has proven to be unbiased in many tests and it is relatively easy to tune with.



Architecture of Proposed model

Advantages:

- These reports are to the investigation of applicability of machine learning techniques for job posting classification.

- Finally, it highlights some observations on future research issues, challenges, and needs.

CHAPTER 2 : LITERATURE SURVEY

General

A literature review is a body of text that aims to review the critical points of current knowledge on and/or methodological approaches to a particular topic. It is secondary sources and discuss published information in a particular subject area and sometimes information in a particular subject area within a certain time period. Its ultimate goal is to bring the reader up to date with current literature on a topic and forms the basis for another goal, such as future research that may be needed in the area and precedes a research proposal and may be just a simple summary of sources. Usually, it has an organizational pattern and combines both summary and synthesis.

A summary is a recap of important information about the source, but a synthesis is a re-organization, reshuffling of information. It might give a new

interpretation of old material or combine new with old interpretations or it might trace the intellectual progression of the field, including major debates. Depending on the situation, the literature review may evaluate the sources and advise the reader on the most pertinent or relevant of them

Review of Literature Survey

Title : Predicting of Job Failure in Compute Cloud Based on Online Extreme Learning Machine: A Comparative Study

Author: CHUNHONG LIU^{1, 2} , JINGJING HAN² , YANLEI SHANG¹ , CHUANCHANG LIU¹ , BO CHENG¹ , AND JUNLIANG CHEN¹

Year: 2017

Early prediction of job failures and specific disposal steps in advance could significantly improve the efficiency of resource utilization in large-scale data center. The existing machine learning-based prediction methods commonly adopt offline working pattern, which cannot be used for online prediction in practical operations in which data arrive sequentially. To solve this problem, a new method based on Online Sequential Extreme Learning Machine (OS-ELM) is proposed in this paper to predict online job termination status. With this method, real-time data are collected according to the sequence of job arriving, the job status could be predicted and the operation model is thus updated based on these data. The method

with online incremental learning strategy has fast learning speed and good generalization. Comparative study using Google trace data shows that prediction accuracy of the proposed method is 93% with updating model in 0.01 seconds. Compared with some state-of-the-art methods such as Support Vector Machine (SVM), Extreme Learning Machine (ELM) and online sequential Support Vector Machine (OS-SVM), the method developed in this paper has many advantages, such as less time-consuming in establishing and updating the model, higher prediction accuracy and precision, and better false negative performance.

Title : Fake Job Recruitment Detection Using Machine Learning Approach

Author: Samir Bandyopadhyay, Shawni Dutta

Year: 2020

To avoid fraudulent post for job in the internet, an automated tool using machine learning based classification techniques is proposed in the paper. Different classifiers are used for checking fraudulent post in the web and the results of those classifiers are compared for identifying the best employment scam detection model. It helps in detecting fake job posts from an enormous number of posts. Two major types of classifiers, such as single classifier and ensemble classifiers are considered for fraudulent job posts detection. However, experimental results indicate that ensemble classifiers are the best classification to detect scams over the single classifiers.

Title : Machine Learning and Job Posting Classification: A Comparative Study

Author: Ibrahim M. Nasser¹ and Amjad H. Alzaanin²

Year: 2020

In this paper, we investigated multiple machine learning classifiers which are, Multinomial Naive Bayes, Support Vector Machine, Decision Tree, K Nearest

Neighbors, and Random Forest in a text classification problem. The data we used contains real and fake job post. We cleaned and pre-processed our data, then we applied TF-IDF for feature extraction. After we implemented the classifiers, we trained and evaluated them. Evaluation metrics used are precision, recall, f-measure, and accuracy. For each classifier, results were summarized and compared with others.

Title : Automatic Detection of Online Recruitment Frauds: Characteristics, Methods, and a Public Dataset

Author: Sokratis Vidros ^{1,* ,†}, Constantinos Kolias ² , Georgios Kambourakis ^{1,2} and Leman Akoglu

Year: 2017

: The critical process of hiring has relatively recently been ported to the cloud. Specifically, the automated systems responsible for completing the recruitment of new employees in an online fashion, aim to make the hiring process more immediate, accurate and cost-efficient. However, the online exposure of such traditional business procedures has introduced new points of failure that may lead to privacy loss for applicants and harm the reputation of organizations. So far, the most common case of Online Recruitment Frauds (ORF), is employment scam. Unlike relevant online fraud problems, the tackling of ORF has not yet received the proper attention, remaining largely unexplored until now. Responding to this need, the work at hand defines and describes the characteristics of this severe and timely novel cyber security research topic. At the same time, it contributes and evaluates the first to our knowledge publicly available dataset of 17,880 annotated job ads, retrieved from the use of a real-life system.

Title : Enhanced RSA Algorithm using Fake Modulus and Fake Public Key Exponent

Author: Raghunandhan K R, Ganesh Aithal, Surendra Shetty, Rakshith N

Year: 2018

In cryptography Public key cryptography plays an important role in the field of data communication. Public key uses two different keys where keys are associated in such a way that, the public key can use to encrypt the data and private key is used to decrypt. RSA is considered as one of the efficient algorithm in public key cryptography. Efficiency of RSA Algorithm mainly depends on how effectively public key components is shared i.e. modulus n and public key exponent e . If these components compromised using mathematical attack, obtaining private key becomes easier job for the intruder. In this paper an enhanced RSA algorithm is proposed to increase the factoring complexity of the public keys, we use fake public key exponent f instead of e and modulus X instead of ' n '. Results of our scheme overcome the limitations of Integer factorization attack. Paper also gives comparative analysis of the proposed work using standard metrics.

CHAPTER 3 : METHODOLOGY

Objectives

The goal is to develop a machine learning model for real or fake job Prediction, to potentially replace the updatable supervised machine learning classification models by predicting results in the form of best accuracy by comparing supervised algorithm.

Project Goals

- Exploration data analysis of variable identification
 - Loading the given dataset
 - Import required libraries packages
 - Analyze the general properties
 - Find duplicate and missing values
 - Checking unique and count values
- Uni-variate data analysis
 - Rename, add data and drop the data
 - To specify data type
- Exploration data analysis of bi-variate and multi-variate
 - Plot diagram of pairplot, heatmap, bar chart and Histogram
- Method of Outlier detection with feature engineering
 - Pre-processing the given dataset
 - Splitting the test and training dataset
 - Comparing the Decision tree and Logistic regression model and random forest etc.

- Comparing algorithm to predict the result
 - Based on the best accuracy

Scope of the Project

The main Scope is to detect the fake job, which is a classic text classification problem with a help of NLP and machine learning algorithm. It is needed to build a model that can differentiate between “Real” job and “Fake” job.

FEASIBILITY STUDY:

Data Wrangling

In this section of the report will load in the data, check for cleanliness, and then trim and clean given dataset for analysis. Make sure that the document steps carefully and justify for cleaning decisions.

Data collection

The data set collected for predicting given data is split into Training set and Test set. Generally, 7:3 ratios are applied to split the Training set and Test set. The Data Model which was created using Random Forest, logistic, Decision tree algorithms and Support vector classifier (SVC) are applied on the Training set and based on the test result accuracy, Test set prediction is done.

Preprocessing

The data which was collected might contain missing values that may lead to inconsistency. To gain better results data need to be preprocessed so as to improve

the efficiency of the algorithm. The outliers have to be removed and also variable conversion need to be done.

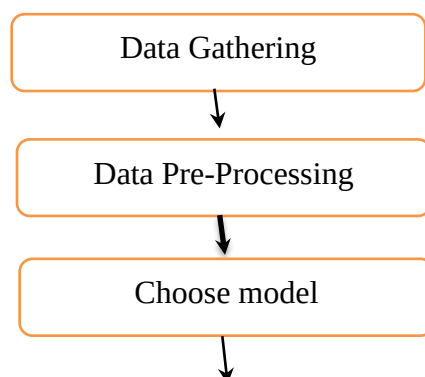
Building the classification model

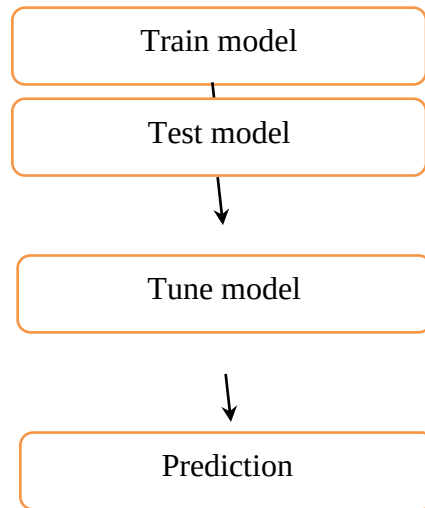
The prediction of wine quality, A high accuracy prediction model is effective because of the following reasons: It provides better results in classification problem.

- It is strong in preprocessing outliers, irrelevant variables, and a mix of continuous, categorical and discrete variables.
- It produces out of bag estimate error which has proven to be unbiased in many tests and it is relatively easy to tune with.

Construction of a Predictive Model

Machine learning needs data gathering have lot of past data's. Data gathering have sufficient historical data and raw data. Before data pre-processing, raw data can't be used directly. It's used to preprocess then, what kind of algorithm with model. Training and testing this model working and predicting correctly with minimum errors. Tuned model involved by tuned time to time with improving the accuracy.





Process of dataflow diagram

LIST OF MODULES:

- ☐ Data Pre-processing
- ☐ Data Analysis of Visualization
- ☐ Comparing Algorithm with prediction in the form of best accuracy result

PROJECT REQUIREMENTS:

General:

Requirements are the basic constrains that are required to develop a system. Requirements are collected while designing the system. The following are the requirements that are to be discussed.

1. Functional requirements
2. Non-Functional requirements

3. Environment requirements

A. Hardware requirements

B. software requirements

Functional requirements:

The software requirements specification is a technical specification of requirements for the software product. It is the first step in the requirements analysis process. It lists requirements of a particular software system. The following details to follow the special libraries like sk-learn, pandas, numpy, matplotlib and seaborn.

Non-Functional Requirements:

Process of functional steps,

1. Problem define
2. Preparing data
3. Evaluating algorithms
4. Improving results
5. Prediction the result

ENVIRONMENTAL REQUIREMENTS:

1. Software Requirements:

Operating System : Windows

Tool : Anaconda with Jupyter Notebook

2. Hardware requirements:

Processor : Pentium IV/III

Hard disk : minimum 80 GB

RAM : minimum 2 GB

SOFTWARE DESCRIPTION

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system “Conda”. The Anaconda distribution is used by over 12 million users and includes more than 1400 popular data-science packages suitable for Windows, Linux, and MacOS. So, Anaconda distribution comes with more than 1,400 packages as well as the Conda package and virtual environment manager called Anaconda Navigator and it eliminates the need to learn to install each library independently.

The open source packages can be individually installed from the Anaconda repository with the conda install command or using the pip install command that is installed with Anaconda. Pip packages provide many of the features of conda packages and in most cases they can work together. Custom packages can be made using the conda build command, and can be shared with others by uploading them to Anaconda Cloud, PyPI or other repositories. The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, you can create new environments that include any version of Python packaged with conda.

ANACONDA NAVIGATOR

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda.org or in a local Anaconda Repository.

Anaconda. Now, if you are primarily doing data science work, Anaconda is also a great option. Anaconda is created by Continuum Analytics, and it is a Python distribution that comes preinstalled with lots of useful python libraries for data science.

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data

processing, predictive analytics, etc.), that aims to simplify package management and deployment.

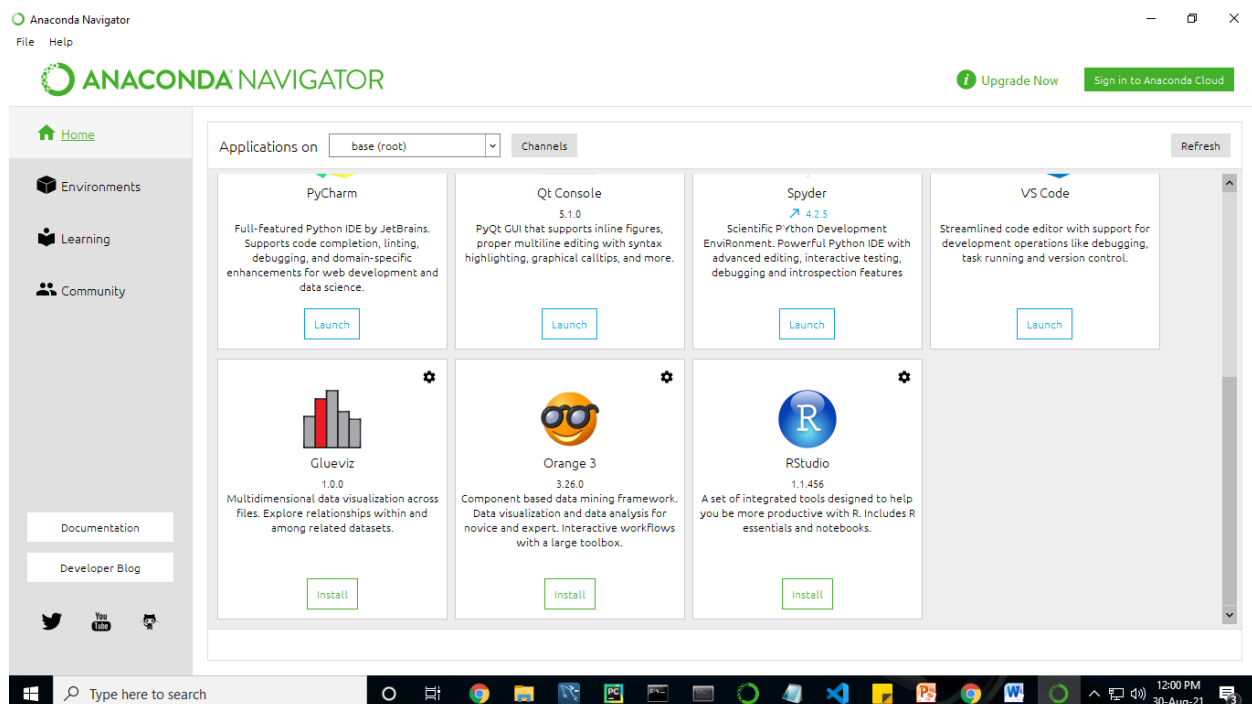
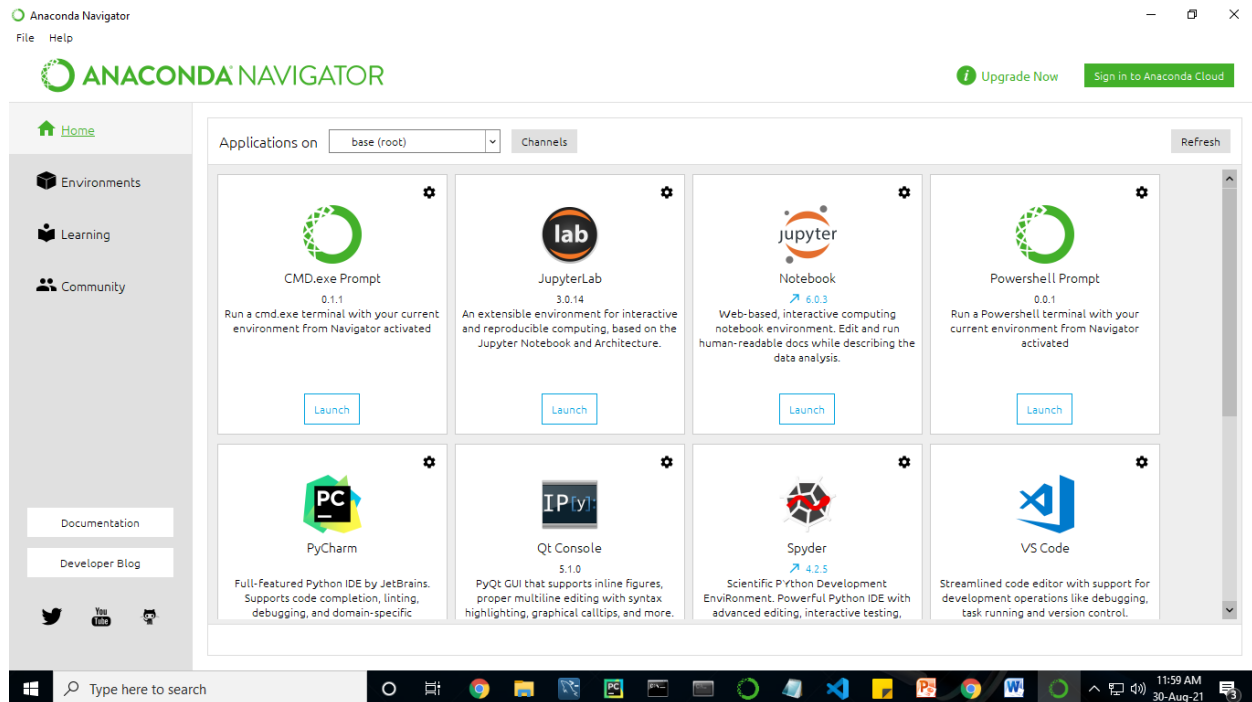
In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages and use multiple environments to separate these different versions.

The command-line program conda is both a package manager and an environment manager. This helps data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages, and update them – all inside Navigator.

The following applications are available by default in Navigator:

- ☐ JupyterLab
- ☐ Jupyter Notebook
- ☐ Spyder
- ☐ PyCharm
- ☐ VSCode
- ☐ Glueviz
- ☐ Orange 3 App
- ☐ RStudio
- ☐ Anaconda Prompt (Windows only)
- ☐ Anaconda PowerShell (Windows only)



Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution.

Navigator allows you to launch common Python programs and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository.

Anaconda comes with many built-in packages that you can easily find with `conda list` on your anaconda prompt. As it has lots of packages (many of which are rarely used), it requires lots of space and time as well. If you have enough space, time and do not want to burden yourself to install small utilities like JSON, YAML, you better go for Anaconda.

Conda :

Conda is an open source, cross-platform, language-agnostic package manager and environment management system that installs, runs, and updates packages and their dependencies. It was created for Python programs, but it can package and distribute software for any language (e.g., R), including multi-language projects. The conda package and environment manager is included in all versions of Anaconda, Miniconda, and Anaconda Repository.

Anaconda is freely available, open source distribution of python and R programming languages which is used for scientific computations. If you are doing any machine learning or deep learning project then this is the best place for you. It consists of many softwares which will help you to build your machine learning project and deep learning project. these softwares have great graphical user interface and these will make your work easy to do. you can also use it to run your python script. These are the software carried by anaconda navigator.

JUPYTER NOTEBOOK

This website acts as “meta” documentation for the Jupyter ecosystem. It has a collection of resources to navigate the tools and communities in this ecosystem, and to help you get started.

Project Jupyter is a project and community whose goal is to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages". It was spun off from IPython in 2014 by Fernando Perez.

Notebook documents are documents produced by the Jupyter Notebook App, which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc...). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc.) as well as executable documents which can be run to perform data analysis.

Installation: The easiest way to install the *Jupyter Notebook App* is installing a scientific python distribution which also includes scientific python packages. The most common distribution is called **Anaconda**

Running the Jupyter Notebook

Launching *Jupyter Notebook App*: The Jupyter Notebook App can be launched by clicking on the *Jupyter Notebook* icon installed by Anaconda in the start menu (Windows) or by typing in a terminal (*cmd* on Windows): “jupyter notebook”

This will launch a new browser window (or a new tab) showing the Notebook Dashboard, a sort of control panel that allows (among other things) to select which notebook to open.

When started, the Jupyter Notebook App can access only files within its start-up folder (including any sub-folder). No configuration is necessary if you place your notebooks in your home folder or subfolders. Otherwise, you need to choose a Jupyter Notebook App start-up folder which will contain all the notebooks.

Save notebooks: Modifications to the notebooks are automatically saved every few minutes. To avoid modifying the original notebook, make a copy of the notebook document (menu file -> make a copy...) and save the modifications on the copy.

Executing a notebook: Download the notebook you want to execute and put it in your notebook folder (or a sub-folder of it).

- ❖ Launch the jupyter notebook app
- ❖ In the Notebook Dashboard navigate to find the notebook: clicking on its name will open it in a new browser tab.

- ❖ Click on the menu *Help* -> *User Interface Tour* for an overview of the Jupyter Notebook App user interface.
- ❖ You can run the notebook document step-by-step (one cell a time) by pressing *shift + enter*.
- ❖ You can run the whole notebook in a single step by clicking on the menu *Cell* -> *Run All*.
- ❖ To restart the kernel (i.e. the computational engine), click on the menu *Kernel* -> *Restart*. This can be useful to start over a computation from scratch (e.g. variables are deleted, open files are closed, etc...).

Purpose: To support interactive data science and scientific computing across all programming languages.

File Extension: An **IPYNB** file is a notebook document created by Jupyter Notebook, an interactive computational environment that helps scientists manipulate and analyze data using Python.

JUPYTER Notebook App:

The *Jupyter Notebook App* is a server-client application that allows editing and running notebook documents via a web browser.

The *Jupyter Notebook App* can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet.

In addition to displaying/editing/running notebook documents, the *Jupyter Notebook App* has a “Dashboard” (Notebook Dashboard), a “control panel” showing local files and allowing to open notebook documents or shutting down their kernels.

kernel: A notebook *kernel* is a “computational engine” that executes the code contained in a Notebook document. The *ipython kernel*, referenced in this guide, executes python code. Kernels for many other languages exist (official kernels).

When you open a Notebook document, the associated *kernel* is automatically launched. When the notebook is *executed* (either cell-by-cell or with menu *Cell -> Run All*), the *kernel* performs the computation and produces the results.

Depending on the type of computations, the *kernel* may consume significant CPU and RAM. Note that the RAM is not released until the *kernel* is shut-down

Notebook Dashboard: The *Notebook Dashboard* is the component which is shown first when you launch Jupyter Notebook App. The *Notebook Dashboard* is mainly used to open notebook documents, and to manage the running kernels (visualize and shutdown).

The *Notebook Dashboard* has other features similar to a file manager, namely navigating folders and renaming/deleting files

Working Process:

- ☐ Download and install anaconda and get the most useful package for machine learning in Python.
- ☐ Load a dataset and understand its structure using statistical summaries and data visualization.
- ☐ Machine learning models, pick the best and build confidence that the accuracy is reliable.

Python is a popular and powerful interpreted language. Unlike R, Python is a complete language and platform that you can use for both research and development and developing production systems. There are also a lot of modules and libraries to choose from, providing multiple ways to do each task. It can feel overwhelming.

The best way to get started using Python for machine learning is to complete a project.

- It will force you to install and start the Python interpreter (at the very least).
- It will give you a bird's eye view of how to step through a small project.
- It will give you confidence, maybe to go on to your own small projects.

When you are applying machine learning to your own datasets, you are working on a project. A machine learning project may not be linear, but it has a number of well-known steps:

- Define Problem.
- Prepare Data.
- Evaluate Algorithms.
- Improve Results.
- Present Results.

The best way to really come to terms with a new platform or tool is to work through a machine learning project end-to-end and cover the key steps. Namely, from loading data, summarizing data, evaluating algorithms and making some predictions.

Here is an overview of what we are going to cover:

1. Installing the Python anaconda platform.
2. Loading the dataset.
3. Summarizing the dataset.
4. Visualizing the dataset.
5. Evaluating some algorithms.
6. Making some predictions.

PYTHON

Introduction:

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a garbage collection system using reference counting. Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible. Python 2 was discontinued with version 2.7.18 in 2020.

Python consistently ranks as one of the most popular programming languages

History:

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to ABC programming language, which was inspired by SETL, capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's Benevolent Dictator For

Life, a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. In January 2019, active Python core developers elected a 5-member "Steering Council" to lead the project. As of 2021, the current members of this council are Barry Warsaw, Brett Cannon, Carol Willing, Thomas Wouters, and Pablo Galindo Salgado.

Python 2.0 was released on 16 October 2000, with many major new features, including a cycle-detecting garbage collector and support for Unicode.

Python 3.0 was released on 3 December 2008. It was a major revision of the language that is not completely backward-compatible. Many of its major features were backported to Python 2.6.x and 2.7.x version series. Releases of Python 3 include the 2 to 3 utility, which automates (at least partially) the translation of Python 2 code to Python 3.

Python 2.7's end-of-life date was initially set at 2015 then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3. No more security patches or other improvements will be released for it. With Python 2's end-of-life, only Python 3.6.x and later are supported.

Python 3.9.2 and 3.8.8 were expedited as all versions of Python (including 2.7) had security issues, leading to possible remote code execution and web cache poisoning.

Design Philosophy & Feature

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by meta-programming and meta-objects (magic

methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python's design offers some support for functional programming in the Lisp tradition. It has filter, map and reduce functions; list comprehensions, dictionaries, sets, and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.

The language's core philosophy is summarized in the document The Zen of Python (PEP 20), which includes aphorisms such as:

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Readability counts.

Rather than having all of its functionality built into its core, Python was designed to be highly extensible (with modules). This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large

standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.

Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to Perl's "there is more than one way to do it" motto, Python embraces a "there should be one— and preferably only one —obvious way to do it" design philosophy. Alex Martelli, a Fellow at the Python Software Foundation and Python book author, writes that "To describe something as 'clever' is not considered a compliment in the Python culture."

Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of the C-Python reference implementation that would offer marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. Cython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter.

Python's developers aim to keep the language fun to use. This is reflected in its name a tribute to the British comedy group Monty Python and in occasionally playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (a reference to a Monty Python sketch) instead of the standard foo and bar.

A common neologism in the Python community is *pythonic*, which can have a wide range of meanings related to program style. To say that code is *pythonic* is to say that it uses Python idioms well, that it is natural or shows fluency in the language, that it conforms with Python's minimalist philosophy and emphasis on

readability. In contrast, code that is difficult to understand or reads like a rough transcription from another programming language is called unpythonic.

Users and admirers of Python, especially those considered knowledgeable or experienced, are often referred to as Pythonistas

Syntax and Semantics :

Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are allowed but are rarely, if ever, used. It has fewer syntactic exceptions and special cases than C or Pascal.

Indentation :

Main article: [Python syntax and semantics & Indentation](#)

Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block. Thus, the program's visual structure accurately represents the program's semantic structure. This feature is sometimes termed the off-side rule, which some other languages share, but in most languages indentation does not have any semantic meaning. The recommended indent size is four spaces.

Statements and control flow :

Python's statements include:

- The assignment statement, using a single equals sign =.
- The if statement, which conditionally executes a block of code, along with else and elif (a contraction of else-if).
- The for statement, which iterates over an iterable object, capturing each element to a local variable for use by the attached block.
- The while statement, which executes a block of code as long as its condition is true.
- The Try statement, which allows exceptions raised in its attached code block to be caught and handled by except clauses; it also ensures that clean-up code in a finally block will always be run regardless of how the block exits.
- The raise statement, used to raise a specified exception or re-raise a caught exception.
- The class statement, which executes a block of code and attaches its local namespace to a class, for use in object-oriented programming.
- The def statement, which defines a function or method.
- The with statement, which encloses a code block within a context manager (for example, acquiring a lock before the block of code is run and releasing the lock afterwards, or opening a file and then closing it), allowing resource-acquisition-is-initialization (RAII) - like behavior and replaces a common try/finally idiom.
- The break statement, exits from a loop.
- The continue statement, skips this iteration and continues with the next item.
- The del statement, removes a variable, which means the reference from the name to the value is deleted and trying to use that variable will cause an error. A deleted variable can be reassigned.

- The pass statement, which serves as a NOP. It is syntactically needed to create an empty code block.
- The assert statement, used during debugging to check for conditions that should apply.
- The yield statement, which returns a value from a generator function and yield is also an operator. This form is used to implement co-routines.
- The return statement, used to return a value from a function.
- The import statement, which is used to import modules whose functions or variables can be used in the current program.

The assignment statement (=) operates by binding a name as a reference to a separate, dynamically-allocated object. Variables may be subsequently rebound at any time to any object. In Python, a variable name is a generic reference holder and does not have a fixed data type associated with it. However, at a given time, a variable will refer to some object, which will have a type. This is referred to as dynamic typing and is contrasted with statically-typed programming languages, where each variable may only contain values of a certain type.

Python does not support tail call optimization or first-class continuations, and, according to Guido van Rossum, it never will.^{[80][81]} However, better support for co-routine-like functionality is provided, by extending Python's generators. Before 2.5, generators were lazy iterators; information was passed uni-directionally out of the generator. From Python 2.5, it is possible to pass information back into a generator function, and from Python 3.3, the information can be passed through multiple stack levels.

Expressions :

Some Python expressions are similar to those found in languages such as C and Java, while some are not:

- Addition, subtraction, and multiplication are the same, but the behavior of division differs. There are two types of divisions in Python. They are floor division (or integer division) `//` and floating-point/division. Python also uses the `**` operator for exponentiation.
- From Python 3.5, the new `@` infix operator was introduced. It is intended to be used by libraries such as NumPy for matrix multiplication.
- From Python 3.8, the syntax `:=`, called the 'walrus operator' was introduced. It assigns values to variables as part of a larger expression.
- In Python, `==` compares by value, versus Java, which compares numerics by value and objects by reference. (Value comparisons in Java on objects can be performed with the `equals()` method.) Python's `is` operator may be used to compare object identities (comparison by reference). In Python, comparisons may be chained, for example `A<=B<=C`.
- Python uses the words `and`, `or`, `not` for its boolean operators rather than the symbolic `&&`, `||`, `!` used in Java and C.
- Python has a type of expression termed a list comprehension as well as a more general expression termed a generator expression.
- Anonymous functions are implemented using `lambda` expressions; however, these are limited in that the body can only be one expression.
- Conditional expressions in Python are written as `x if c else y` (different in order of operands from the `c ? x : y` operator common to many other languages).

- Python makes a distinction between lists and tuples. Lists are written as [1, 2, 3], are mutable, and cannot be used as the keys of dictionaries (dictionary keys must be immutable in Python). Tuples are written as (1, 2, 3), are immutable and thus can be used as the keys of dictionaries, provided all elements of the tuple are immutable. The + operator can be used to concatenate two tuples, which does not directly modify their contents, but rather produces a new tuple containing the elements of both provided tuples. Thus, given the variable t initially equal to (1, 2, 3), executing t = t + (4, 5) first evaluates t + (4, 5), which yields (1, 2, 3, 4, 5), which is then assigned back to t, thereby effectively "modifying the contents" of t, while conforming to the immutable nature of tuple objects. Parentheses are optional for tuples in unambiguous contexts.
- Python features sequence unpacking wherein multiple expressions, each evaluating to anything that can be assigned to (a variable, a writable property, etc.), are associated in an identical manner to that forming tuple literals and, as a whole, are put on the left-hand side of the equal sign in an assignment statement. The statement expects an iterable object on the right-hand side of the equal sign that produces the same number of values as the provided writable expressions when iterated through and will iterate through it, assigning each of the produced values to the corresponding expression on the left.
- Python has a "string format" operator %. This functions analogously to printf format strings in C, e.g. "spam=%s eggs=%d" % ("blah",2) evaluates to "spam=blah eggs=2". In Python 3 and 2.6+, this was supplemented by the format() method of the str class, e.g. "spam={0} eggs={1}".format("blah",2). Python 3.6 added "f-strings": blah = "blah"; eggs = 2; f'spam={blah} eggs={eggs}'
- Strings in Python can be concatenated, by "adding" them (same operator as for adding integers and floats). E.g. "spam" + "eggs" returns "spameggs". Even if

your strings contain numbers, they are still added as strings rather than integers. E.g. `"2" + "2"` returns `"2"`.

- Python has various kinds of string literals:
 - Strings delimited by single or double quote marks. Unlike in Unix shells, Perl and Perl-influenced languages, single quote marks and double quote marks function identically. Both kinds of string use the backslash (`\`) as an escape character. String interpolation became available in Python 3.6 as "formatted string literals".
 - Triple-quoted strings, which begin and end with a series of three single or double quote marks. They may span multiple lines and function like here documents in shells, Perl and Ruby.
 - Raw string varieties, denoted by prefixing the string literal with an `r`. Escape sequences are not interpreted; hence raw strings are useful where literal backslashes are common, such as regular expressions and Windows-style paths. Compare `"@-quoting"` in C#.
- Python has array index and array slicing expressions on lists, denoted as `a[Key]`, `a[start:stop]` or `a[start:stop:step]`. Indexes are zero-based, and negative indexes are relative to the end. Slices take elements from the start index up to, but not including, the stop index. The third slice parameter, called step or stride, allows elements to be skipped and reversed. Slice indexes may be omitted, for example `a[:]` returns a copy of the entire list. Each element of a slice is a shallow copy.

In Python, a distinction between expressions and statements is rigidly enforced, in contrast to languages such as Common Lisp, Scheme, or Ruby. This leads to duplicating some functionality. For example:

- List comprehensions vs. for-loops
- Conditional expressions vs. if blocks
- The `eval()` vs. `exec()` built-in functions (in Python 2, `exec` is a statement); the former is for expressions, the latter is for statements.

Statements cannot be a part of an expression, so list and other comprehensions or lambda expressions, all being expressions, cannot contain statements. A particular case of this is that an assignment statement such as `a=1` cannot form part of the conditional expression of a conditional statement. This has the advantage of avoiding a classic C error of mistaking an assignment operator `=` for an equality operator `==` in conditions: `if (c==1) {...}` is syntactically valid (but probably unintended) C code but `if c=1: ...` causes a syntax error in Python.

Methods :

Methods on objects are functions attached to the object's class; the syntax `instance.method(argument)` is, for normal methods and functions, syntactic sugar for `Class.method(instance, argument)`. Python methods have an explicit `self` parameter access instance data, in contrast to the implicit `self` (or `this`) in some other object-oriented programming languages (e.g., C++, Java, Objective-C, or Ruby). Apart from this Python also provides methods, sometimes called *d-under methods* due to their names beginning and ending with double-underscores, to extend the functionality of custom class to support native functions such as `print`, `length`, `comparison`, support for arithmetic operations, type conversion, and many more.

Typing :

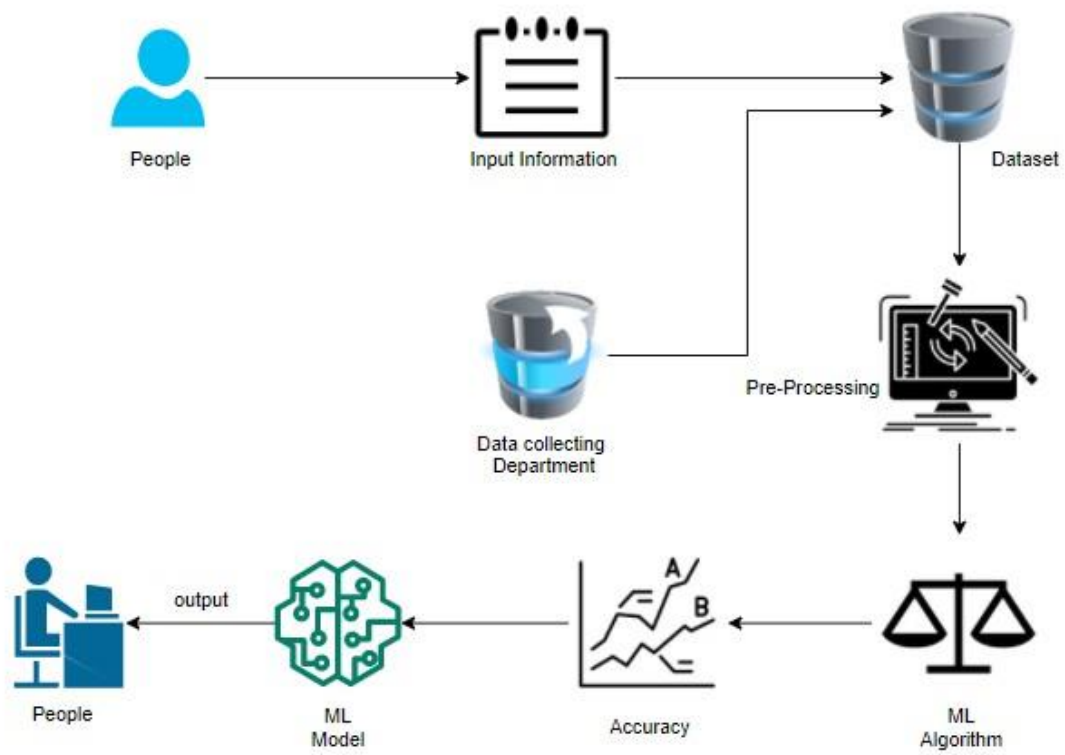
Python uses duck typing and has typed objects but untyped variable names. Type constraints are not checked at compile time; rather, operations on an object may fail, signifying that the given object is not of a suitable type. Despite being dynamically-typed, Python is strongly-typed, forbidding operations that are not well-defined (for example, adding a number to a string) rather than silently attempting to make sense of them.

Python allows programmers to define their own types using classes, which are most often used for object-oriented programming. New instances of classes are constructed by calling the class (for example, `SpamClass()` or `EggsClass()`), and the classes are instances of the metaclass type (itself an instance of itself), allowing meta-programming and reflection.

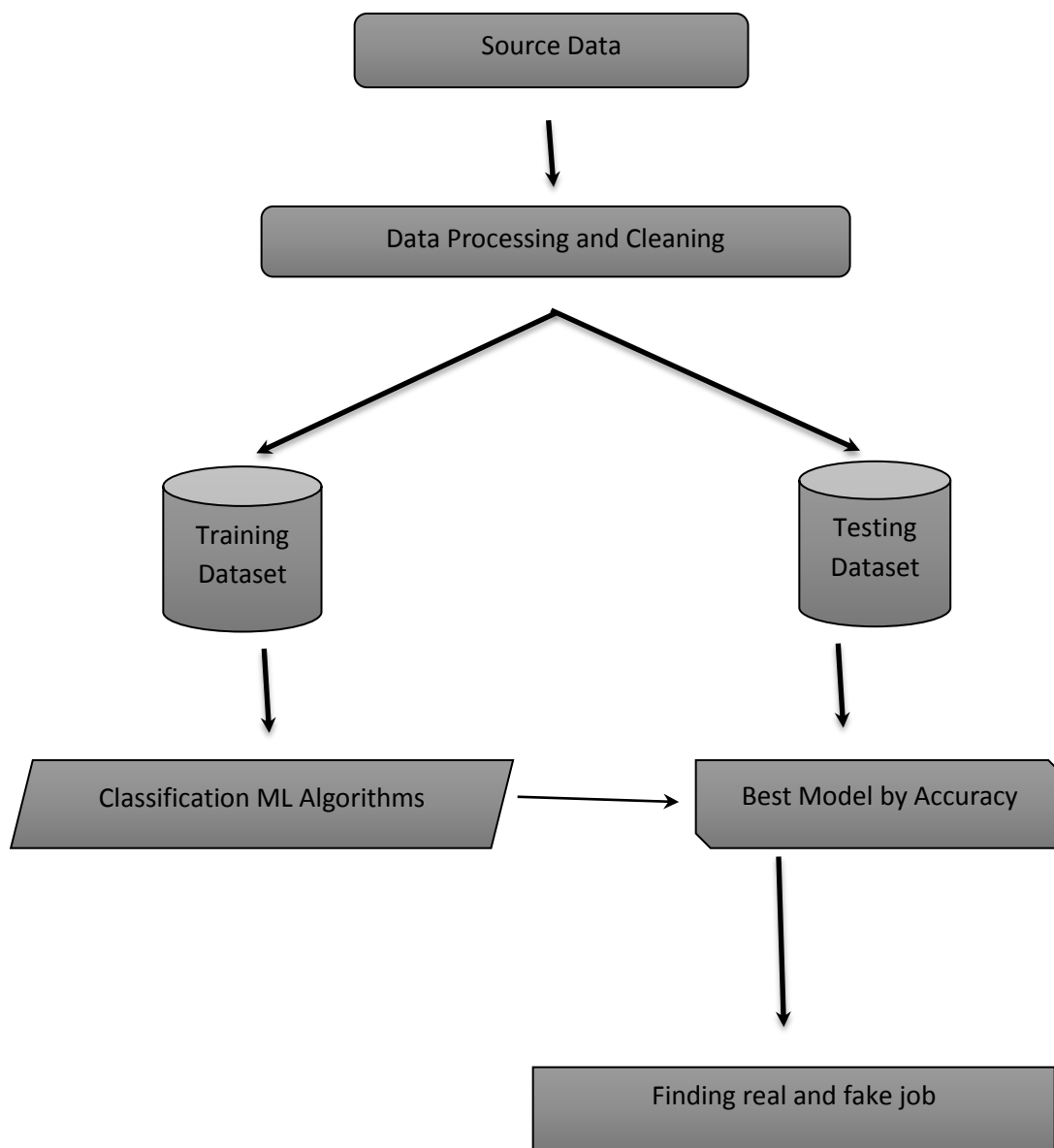
Before version 3.0, Python had two kinds of classes: old-style and new-style. The syntax of both styles is the same, the difference being whether the class object is inherited from, directly or indirectly (all new-style classes inherit from `object` and are instances of `type`). In versions of Python 2 from Python 2.2 onwards, both kinds of classes can be used. Old-style classes were eliminated in Python 3.0.

The long-term plan is to support gradual typing and from Python 3.5, the syntax of the language allows specifying static types but they are not checked in the default implementation, CPython. An experimental optional static type checker named `mypy` supports compile-time type checking.

SYSTEM ARCHITECTURE:

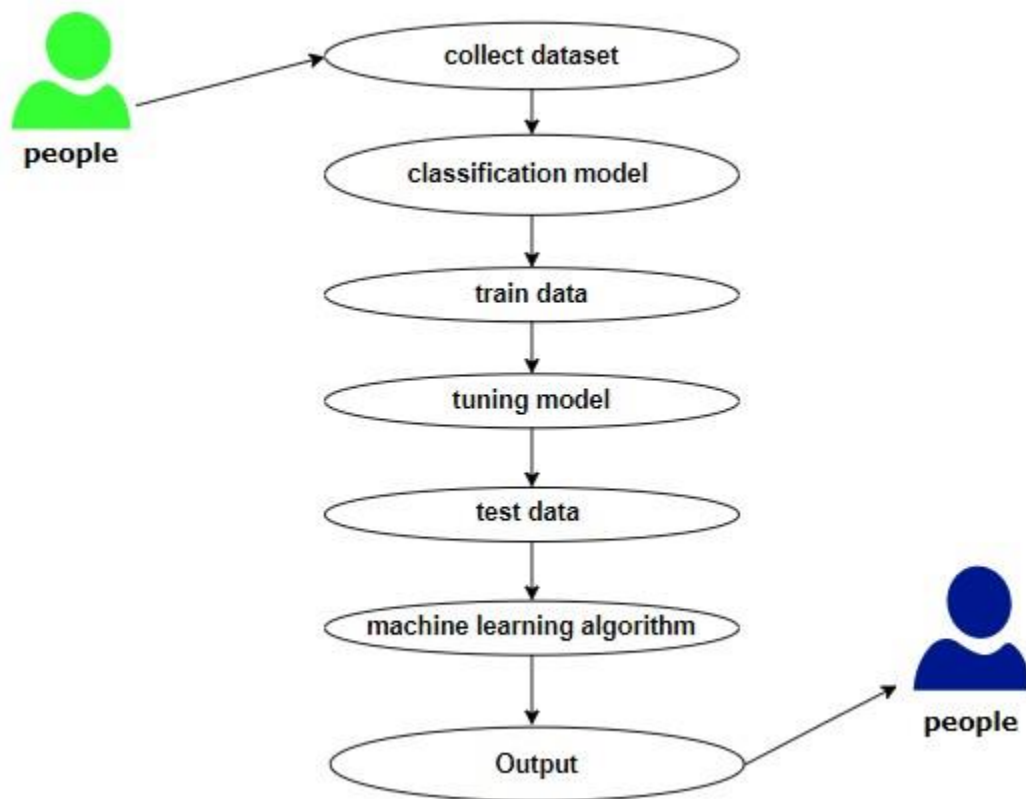


WORK FLOW DIAGRAM:



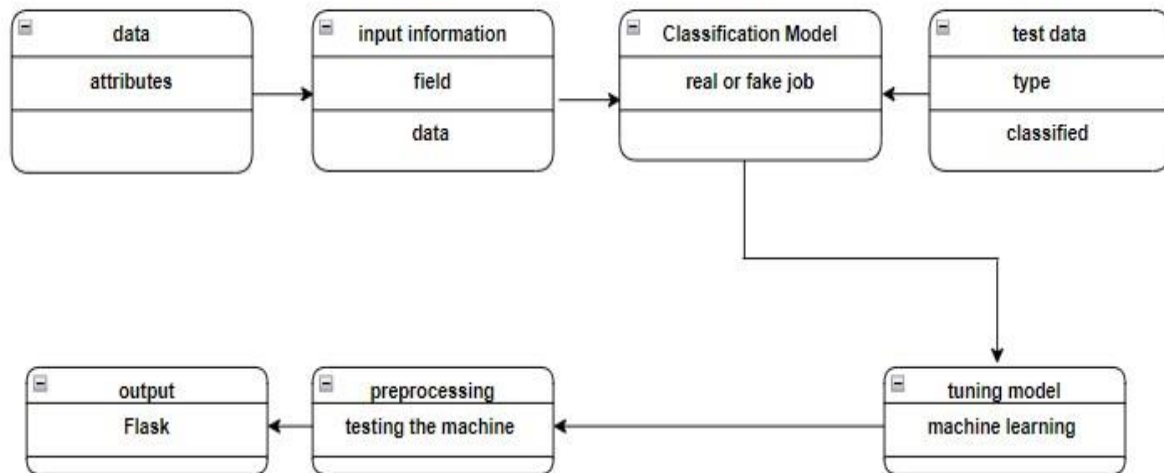
Workflow Diagram

USE CASE DIAGRAM:



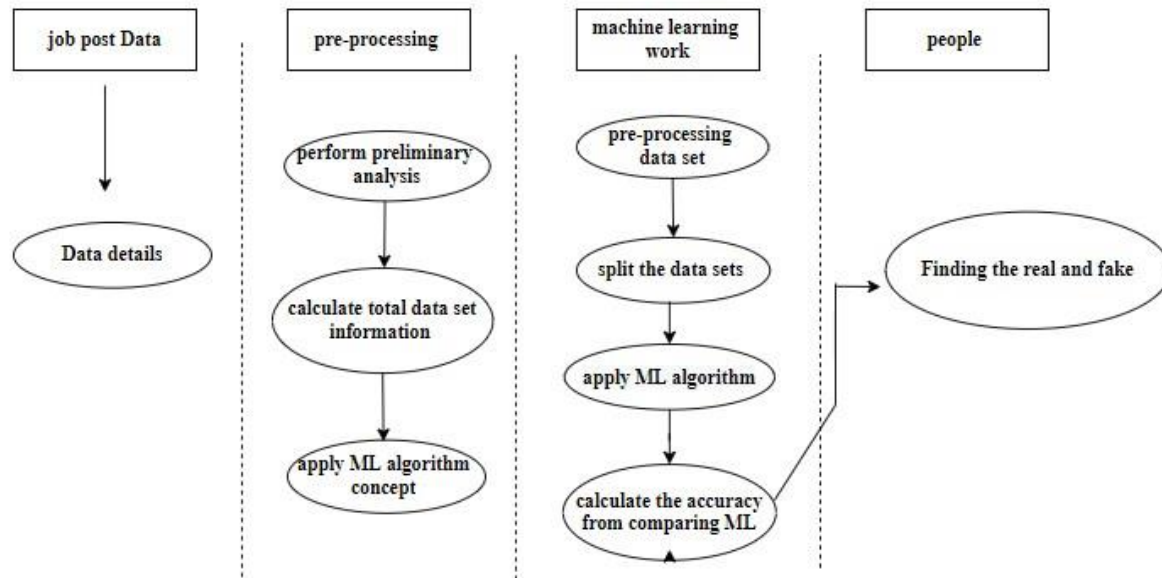
Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases. So, it can say that uses cases are nothing but the system functionalities written in an organized manner.

CLASS DIAGRAM:



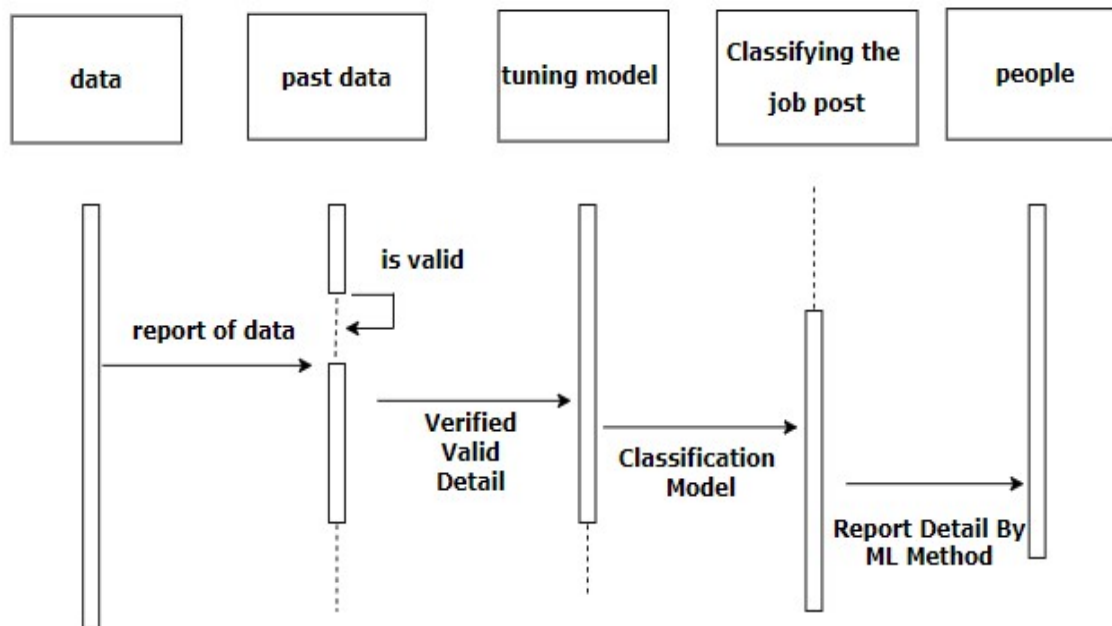
Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So a collection of class diagrams represent the whole system. The name of the class diagram should be meaningful to describe the aspect of the system. Each element and their relationships should be identified in advance Responsibility (attributes and methods) of each class should be clearly identified for each class minimum number of properties should be specified and because, unnecessary properties will make the diagram complicated. Use notes whenever required to describe some aspect of the diagram and at the end of the drawing it should be understandable to the developer/coder. Finally, before making the final version, the diagram should be drawn on plain paper and rework as many times as possible to make it correct.

ACTIVITY DIAGRAM:



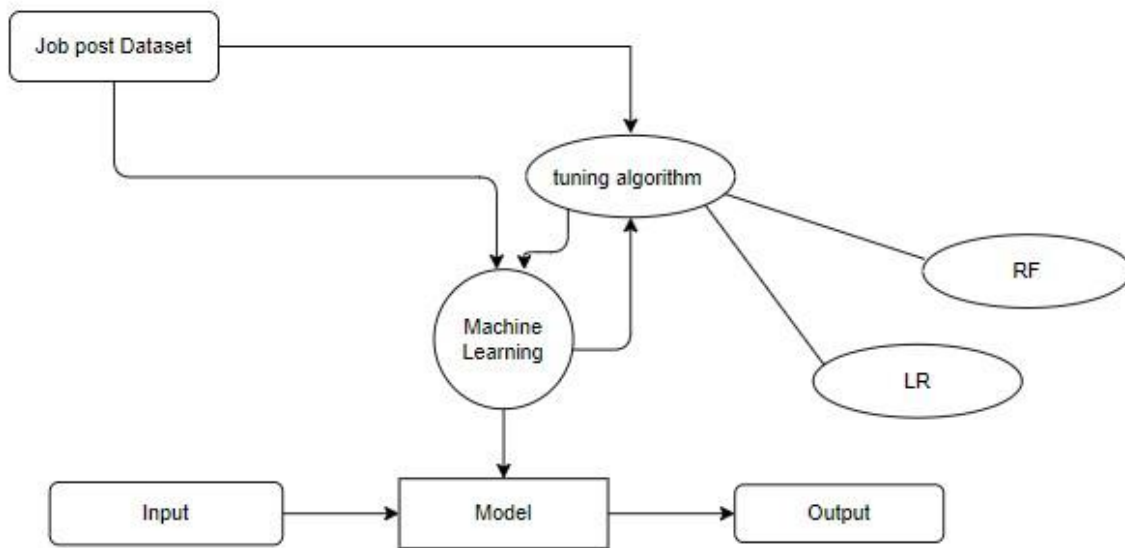
Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part. It does not show any message flow from one activity to another. Activity diagram is some time considered as the flow chart. Although the diagrams looks like a flow chart but it is not. It shows different flow like parallel, branched, concurrent and single.

SEQUENCE DIAGRAM:



Sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modeling, which focuses on identifying the behavior within your system. Other dynamic modeling techniques include activity diagramming, communication diagramming, timing diagramming, and interaction overview diagramming. Sequence diagrams, along with class diagrams and physical data models are in my opinion the most important design-level models for modern business application development.

ENTITY RELATIONSHIP DIAGRAM (ERD)



An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation of an information system that depicts the relationships among people, objects, places, concepts or events within that system. An ERD is a data modeling technique that can help define business processes and be used as the foundation for a relational database. Entity relationship diagrams provide a visual starting point for database design that can also be used to help determine information system requirements throughout an organization. After a relational database is rolled out, an ERD can still serve as a referral point, should any debugging or business process re-engineering be needed later.

CHAPTER 4 : RESULT AND DISCUSSION

The machine learning approach is developed in this model to predict the real or fake job posting. To view the dependent and independent variable it is preprocessed and examined, and to get the results various machine learning algorithms are used. Dataset are combined to form a common dataset and these would be used to get the pattern and result with higher accuracy. The data will be loaded, cleaned and trimmed for result.

DISCUSSION

Data Pre-processing

Validation techniques in machine learning are used to get the error rate of the Machine Learning (ML) model, which can be considered as close to the true error rate of the dataset. If the data volume is large enough to be representative of the population, you may not need the validation techniques. However, in real-world scenarios, to work with samples of data that may not be a true representative of the population of given dataset. To finding the missing value, duplicate value and description of data type whether it is float variable or integer. The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyper parameters.

The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration. The validation set is used to evaluate a given model, but this is for frequent evaluation. It as machine learning engineers use this data to fine-tune the model hyper parameters. Data collection, data analysis, and the process of addressing data content, quality, and structure can add up to a time-consuming to-do list. During the process of data identification, it helps to understand your data and its properties; this knowledge will help you choose which algorithm to use to build your model.

A number of different data cleaning tasks using Python's Pandas library and specifically, it focus on probably the biggest data cleaning task, missing values and

it able to more quickly clean data. It wants to spend less time cleaning data, and more time exploring and modeling.

Some of these sources are just simple random mistakes. Other times, there can be a deeper reason why data is missing. It's important to understand these different types of missing data from a statistics point of view. The type of missing data will influence how to deal with filling in the missing values and to detect missing values, and do some basic imputation and detailed statistical approach for dealing with missing data. Before, joint into code, it's important to understand the sources of missing data. Here are some typical reasons why data is missing:

- User forgot to fill in a field.
- Data was lost while transferring manually from a legacy database.
- There was a programming error.
- Users chose not to fill out a field tied to their beliefs about how the results would be used or interpreted.

Variable identification with Uni-variate, Bi-variate and Multi-variate analysis:

- ☐ import libraries for access and functional purpose and read the given dataset
- ☐ General Properties of Analyzing the given dataset
- ☐ Display the given dataset in the form of data frame
- ☐ show columns
- ☐ shape of the data frame
- ☐ To describe the data frame

- ☐ Checking data type and information about dataset
- ☐ Checking for duplicate data
- ☐ Checking Missing values of data frame
- ☐ Checking unique values of data frame
- ☐ Checking count values of data frame
- ☐ Rename and drop the given data frame
- ☐ To specify the type of values
- ☐ To create extra columns

Data Validation/ Cleaning/Preparing Process

Importing the library packages with loading given dataset. To analyzing the variable identification by data shape, data type and evaluating the missing values, duplicate values. A validation dataset is a sample of data held back from training your model that is used to give an estimate of model skill while tuning model's and procedures that you can use to make the best use of validation and test datasets when evaluating your models. Data cleaning / preparing by rename the given dataset and drop the column etc. to analyze the uni-variate, bi-variate and multi-variate process. The steps and techniques for data cleaning will vary from dataset to dataset. The primary goal of data cleaning is to detect and remove errors and anomalies to increase the value of data in analytics and decision making.

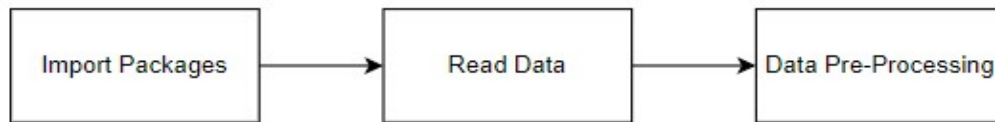
	Unnamed: 0	job_id	telecommuting	has_company_logo	has_questions
count	1404.000000	1404.000000	1404.000000	1404.000000	1404.000000
mean	701.500000	7967.114672	0.128205	0.965100	0.687322
std	405.444201	4862.817023	0.334437	0.183593	0.463750
min	0.000000	7.000000	0.000000	0.000000	0.000000
25%	350.750000	4403.000000	0.000000	1.000000	0.000000
50%	701.500000	7091.000000	0.000000	1.000000	1.000000
75%	1052.250000	11740.000000	0.000000	1.000000	1.000000
max	1403.000000	17866.000000	1.000000	1.000000	1.000000

```

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1404 entries, 0 to 1403
Data columns (total 19 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Unnamed: 0                            1404 non-null   int64
1   job_id                                1404 non-null   int64
2   title                                 1404 non-null   object
3   location                              1404 non-null   object
4   department                            1404 non-null   object
5   salary_range                          1404 non-null   object
6   company_profile                       1404 non-null   object
7   description                           1404 non-null   object
8   requirements                          1404 non-null   object
9   benefits                              1404 non-null   object
10  telecommuting                         1404 non-null   int64
11  has_company_logo                      1404 non-null   int64
12  has_questions                         1404 non-null   int64
13  employment_type                       1404 non-null   object
14  required_experience                    1404 non-null   object
15  required_education                    1404 non-null   object
16  industry                              1404 non-null   object
17  function                              1404 non-null   object
18  fraudulent                            1404 non-null   object
dtypes: int64(5), object(14)
memory usage: 219.4+ KB

```

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

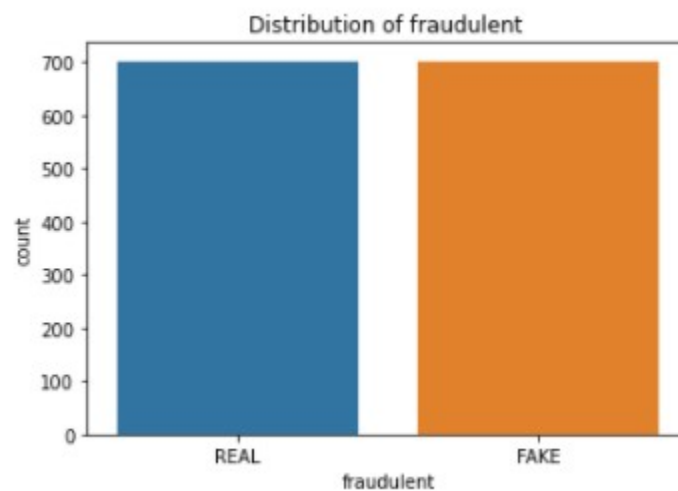
output : removing noisy data

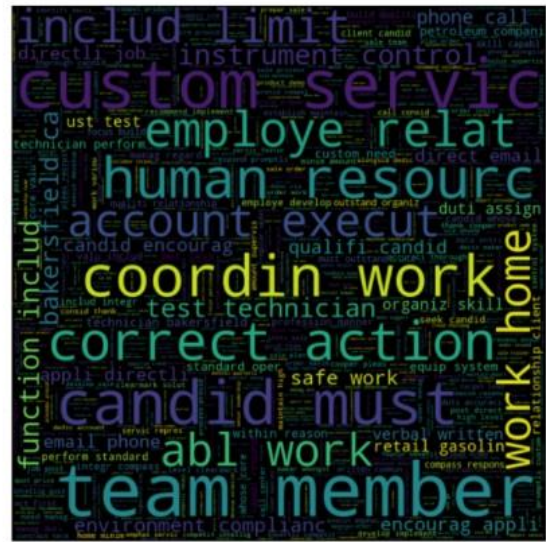
Exploration data analysis of visualization

Data visualization is an important skill in applied statistics and machine learning. Statistics does indeed focus on quantitative descriptions and estimations of data. Data visualization provides an important suite of tools for gaining a qualitative understanding. This can be helpful when exploring and getting to know a dataset and can help with identifying patterns, corrupt data, outliers, and much more. With a little domain knowledge, data visualizations can be used to express and demonstrate key relationships in plots and charts that are more visceral and stakeholders than measures of association or significance. Data visualization and exploratory data analysis are whole fields themselves and it will recommend a deeper dive into some the books mentioned at the end.

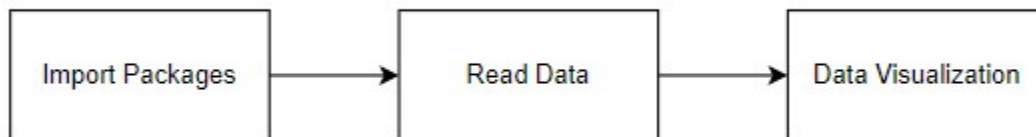
Sometimes data does not make sense until it can look at in a visual form, such as with charts and plots. Being able to quickly visualize of data samples and others is an important skill both in applied statistics and in applied machine learning. It will discover the many types of plots that you will need to know when visualizing data in Python and how to use them to better understand your own data.

- How to chart time series data with line plots and categorical quantities with bar charts.
- How to summarize data distributions with histograms and box plots.





MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

output : visualized data

Comparing Algorithm with prediction in the form of best accuracy result

It is important to compare the performance of multiple different machine learning algorithms consistently and it will discover to create a test harness to compare multiple different machine learning algorithms in Python with scikit-learn. It can use this test harness as a template on your own machine learning problems and add more and different algorithms to compare. Each model will have different performance characteristics. Using resampling methods like cross validation, you can get an estimate for how accurate each model may be on unseen data. It needs to be able to use these estimates to choose one or two best models from the suite of models that you have created. When have a new dataset, it is a good idea to visualize the data using different techniques in order to look at the data from different perspectives. The same idea applies to model selection. You should use a number of different ways of looking at the estimated accuracy of your machine learning algorithms in order to choose the one or two to finalize. A way to do this is to use different visualization methods to show the average accuracy, variance and other properties of the distribution of model accuracies.

In the next section you will discover exactly how you can do that in Python with scikit-learn. The key to a fair comparison of machine learning algorithms is ensuring that each algorithm is evaluated in the same way on the same data and it can achieve this by forcing each algorithm to be evaluated on a consistent test harness.

In the example below 4 different algorithms are compared:

- Logistic Regression

□ Random Forest

The K-fold cross validation procedure is used to evaluate each algorithm, importantly configured with the same random seed to ensure that the same splits to the training data are performed and that each algorithm is evaluated in precisely the same way. Before that comparing algorithm, Building a Machine Learning Model using install Scikit-Learn libraries. In this library package have to done preprocessing, linear model with logistic regression method, cross validating by KFold method, ensemble with random forest method and tree with decision tree classifier. Additionally, splitting the train set and test set. To predicting the result by comparing accuracy.

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. To achieving better results from the applied model in Machine Learning method of the data has to be in a proper manner. Some specified Machine Learning model needs information in a specified format, for example, Random Forest algorithm does not support null values. Therefore, to execute random forest algorithm null values have to be managed from the original raw data set. And another aspect is that data set should be formatted in such a way that more than one Machine Learning and Deep Learning algorithms are executed in given dataset.

Prediction result by accuracy:

Logistic regression algorithm also uses a linear equation with independent predictors to predict a value. The predicted value can be anywhere between

negative infinity to positive infinity. It need the output of the algorithm to be classified variable data. Higher accuracy predicting result is logistic regression model by comparing the best accuracy.

False Positives (FP): A person who will pay predicted as defaulter. When actual class is no and predicted class is yes. E.g. if actual class says this passenger did not survive but predicted class tells you that this passenger will survive.

False Negatives (FN): A person who default predicted as payer. When actual class is yes but predicted class in no. E.g. if actual class value indicates that this passenger survived and predicted class tells you that passenger will die.

True Positives (TP): A person who will not pay predicted as defaulter. These are the correctly predicted positive values which means that the value of actual class is yes and the value of predicted class is also yes. E.g. if actual class value indicates that this passenger survived and predicted class tells you the same thing.

True Negatives (TN): A person who default predicted as payer. These are the correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no. E.g. if actual class says this passenger did not survive and predicted class tells you the same thing.

$$\text{True Positive Rate(TPR)} = \text{TP} / (\text{TP} + \text{FN})$$

$$\text{False Positive rate(FPR)} = \text{FP} / (\text{FP} + \text{TN})$$

Accuracy: The Proportion of the total number of predictions that is correct otherwise overall how often the model predicts correctly defaulters and non-defaulters.

Accuracy calculation:

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN})$$

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best. Yes, accuracy is a great measure but only when you have symmetric datasets where values of false positive and false negatives are almost same.

Precision: The proportion of positive predictions that are actually correct.

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. The question that this metric answer is of all passengers that labeled as survived, how many actually survived? High precision relates to the low false positive rate. We have got 0.788 precision which is pretty good.

Recall: The proportion of positive observed values correctly predicted. (The proportion of actual defaulters that the model will correctly predict)

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

Recall(Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes.

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall.

General Formula:

$$\text{F-Measure} = 2\text{TP} / (2\text{TP} + \text{FP} + \text{FN})$$

F1-Score Formula:

$$\text{F1 Score} = 2 * (\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$$

ALGORITHM AND TECHNIQUES

Algorithm Explanation

In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam

or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc. In Supervised Learning, algorithms learn from labeled data. After understanding the data, the algorithm determines which label should be given to new data based on pattern and associating the patterns to the unlabeled new data.

Used Python Packages:

sklearn:

- In python, sklearn is a machine learning package which include a lot of ML algorithms.
- Here, we are using some of its modules like `train_test_split`, `DecisionTreeClassifier` or `Logistic Regression` and `accuracy_score`.

NumPy:

- It is a numeric python module which provides fast maths functions for calculations.
- It is used to read data in numpy arrays and for manipulation purpose.

Pandas:

- Used to read and write different files.
- Data manipulation can be done easily with data frames.

Matplotlib:

- Data visualization is a useful way to help with identify the patterns from given dataset.

- Data manipulation can be done easily with data frames.

Natural Language Processing (NLP):

Natural language processing (NLP) allows machines to read and understand human language. A sufficiently powerful natural language processing system would enable natural-language user interfaces and the acquisition of knowledge directly from human-written sources, such as newswire texts. Some straightforward applications of natural language processing include information retrieval, text mining, question answering and machine translation. Many current approaches use word co-occurrence frequencies to construct syntactic representations of text. “Keyword spotting” strategies for search are popular and scalable but dumb; a search query for “dog” might only match documents with the literal word “dog” and miss a document with the word “poodle”. “Lexical affinity” strategies use the occurrence of words such as “accident” to assess the sentiment of a document. Modern statistical NLP approaches can combine all these strategies as well as others, and often achieve acceptable accuracy at the page or paragraph level. Beyond semantic NLP, the ultimate goal of “narrative” NLP is to embody a full understanding of commonsense reasoning.

1. Remove stopwords: There are a few words which are very commonly used when humans interact, but these words don’t make any sense or add any extra value. Additionally, there might be few words which are not required for the business case given in hand. So, these words need to be deleted from the data.

The NLTK package has a defined set of stopwords for different languages like English. Here, we will focus on ‘english’ stopwords. One can also consider additional stopwords if required.

2.Tokenization : This is one of the common practices while working on text data. This helps to split a phrase, sentence, or paragraph into small units like words or terms. Each unit is called a token. There are different types of tokenization. We have already used this in above examples for stemming, POS tagging, and NER. Below are different ways to tokenize the text.

Vectorization/Word Embedding

Once cleaning and tokenization is done, extracting features from the clean data is very important as the machine doesn't understand the words but numbers.

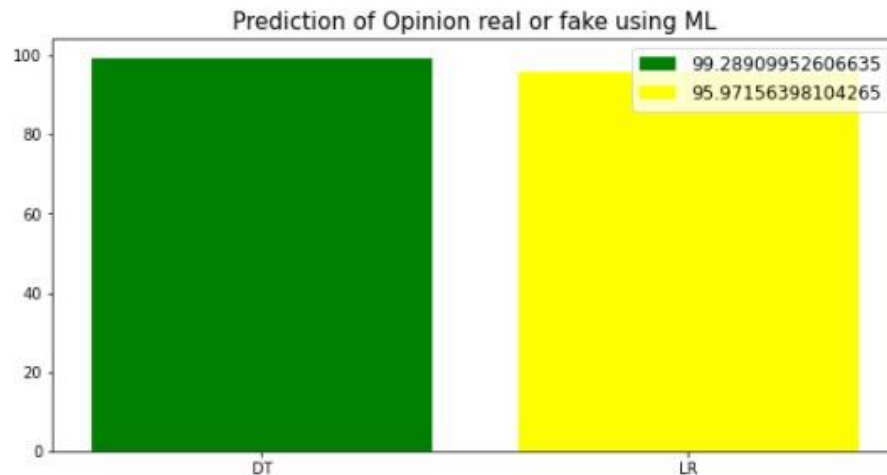
Vectorization helps to map the words to a vector of real numbers, which further helps into predictions. This helps to extract the important features.

Logistic Regression

It is a statistical method for analysing a data set in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes). The goal of logistic regression is to find the best fitting model to describe the relationship between the dichotomous characteristic of interest (dependent variable = response or outcome variable) and a set of independent (predictor or explanatory) variables. Logistic regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.).

In other words, the logistic regression model predicts $P(Y=1)$ as a function of X . Logistic regression Assumptions:

- ☐ Binary logistic regression requires the dependent variable to be binary.
- ☐ For a binary regression, the factor level 1 of the dependent variable should represent the desired outcome.
- ☐ Only the meaningful variables should be included.
- ☐ The independent variables should be independent of each other. That is, the model should have little.
- ☐ The independent variables are linearly related to the log odds.
- ☐ Logistic regression requires quite large sample sizes.



Accuracy result of Random Forest Classifier is: 99.28909952606635

Classification report of Random Forest Classifier : Results:

	precision	recall	f1-score	support
0	1.00	0.99	0.99	211
1	0.99	1.00	0.99	211
accuracy			0.99	422
macro avg	0.99	0.99	0.99	422
weighted avg	0.99	0.99	0.99	422

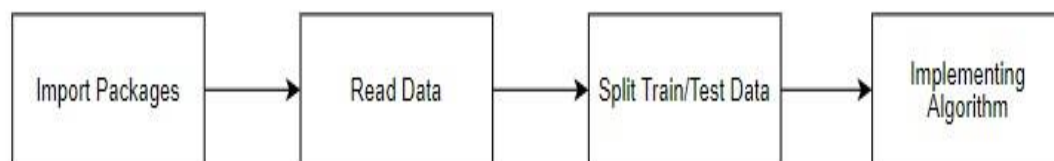
Confusion Matrix result of Random Forest Classifier : is:

```
[[208  3]
 [  0 211]]
```

Sensitivity : 0.985781990521327

Specificity : 1.0

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

Random Forest Classifier

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of over fitting to their training set. Random forest is a type of supervised machine learning algorithm based on ensemble learning. Ensemble learning is a type of learning where you join different types of algorithms or same algorithm multiple times to form a more powerful prediction model. The random forest algorithm combines multiple algorithm of the same type i.e. multiple decision *trees*, resulting in a *forest of trees*, hence the name "Random Forest". The random forest algorithm can be used for both regression and classification tasks.

The following are the basic steps involved in performing the random forest algorithm:

- ☐ Pick N random records from the dataset.
- ☐ Build a decision tree based on these N records.
- ☐ Choose the number of trees you want in your algorithm and repeat steps 1 and 2.

In case of a regression problem, for a new record, each tree in the forest predicts a value for Y (output). The final value can be calculated by taking the average of all the values predicted by all the trees in forest. Or, in case of a classification problem, each tree in the forest predicts the category to which the new record belongs. Finally, the new record is assigned to the category that wins the majority vote.

```

In [16]: from sklearn.feature_extraction.text import CountVectorizer
        from sklearn.model_selection import train_test_split

In [17]: cv = CountVectorizer()
        X = cv.fit_transform(x) # Fit the Data
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

In [18]: from sklearn.linear_model import LogisticRegression
        clf = LogisticRegression()
        clf.fit(X_train, y_train)

In [21]: input_word=input("ENTER THE SENTENCE:")

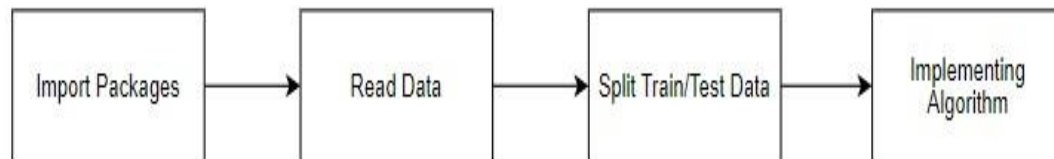
ENTER THE SENTENCE:Qualified candidates are encouraged to apply directly to this job posting. A Direct email and phone calls are not being considered. Thank you for your cooperation. A Please no recruiters. A UST Testing Technician II Bakersfield, CA A Local Petroleum Company operates primarily in retail and wholesale of motor fuels and other related petroleum products and is seeking talented, experienced, maintenance technicians to perform troubleshooting and maintenance on their retail gasoline equipment. A RESPONSIBILITIES The ideal candidate will have experience working with retail gasoline dispensing and peripheral equipment, UST systems, weights and measures compliance, and computer applications. Certifications with Gilbarco, Ruby, Sapphire, and VederRoot Tank Gauging preferred. Schedule testing with local agencies. A

In [22]: data = cv.transform([input_word]).toarray()
        print(clf.predict(data))

['FAKE']

```

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

CHAPTER 5 : CONCLUSION AND FUTURE WORK

Conclusion

The analytical process started from data cleaning and processing, missing value, exploratory analysis and finally model building and evaluation. The best accuracy on public test set is higher accuracy score will be find out. This application can help to find the Prediction of Real and fake job.

Future Work

- ☐ Real and fake job prediction to connect with cloud.
- ☐ To optimize the work to implement in Artificial Intelligence environment.

APPENDICES

A.SAMPLE CODE

Module – 1

Pre-Processing

```
#import library packages
```

```
import pandas as pd
```

```
import numpy as np
```

In []:

```
import warnings
```

```
warnings.filterwarnings("ignore")
```

In []:

```
#Load given dataset
```

```
data = pd.read_csv("r_or_f.csv")
```

Before drop the given dataset:

In []:

```
data.head()
```

In []:

```
#shape
```

```
data.shape
```

After drop the given dataset:

In []:

```
df = data.dropna()
```

In []:

```
df.head()
```

In []:

```
#shape
```

```
df.shape
```

In []:

```
#columns
```

```
df.columns
```

```
In [ ]:
```

```
#To describe the dataframe
```

```
df.describe()
```

```
In [ ]:
```

```
#Checking datatype and information about dataset
```

```
df.info()
```

```
Checking duplicate values of dataframe
```

```
In [ ]:
```

```
#Checking for duplicate data
```

```
df.duplicated()
```

```
In [ ]:
```

```
sum(df.duplicated())
```

```
In [ ]:
```

```
#Checking sum of missing values
```

```
df.isnull().sum()
```

```
In [ ]:
```

```
df.fraudulent.unique()
```

```
In [ ]:
```

```
df.description.unique()
```

```
In [ ]:
```

```
from sklearn.preprocessing import LabelEncoder
```

```
var=['Unnamed: 0', 'job_id', 'title', 'location', 'department',  
     'salary_range', 'company_profile', 'description', 'requirements',  
     'benefits', 'telecommuting', 'has_company_logo', 'has_questions',  
     'employment_type', 'required_experience', 'required_education',
```

```

        'industry', 'function', 'fraudulent']
le=LabelEncoder()
for i in var:
    df[i]=le.fit_transform(df[i]).astype(int)

```

In []:

```
df.head()
```

Module – 2

Visualization

```

#import library packages
import pandas as p
import matplotlib.pyplot as plt
import seaborn as s
import numpy as n

```

In []:

```

import warnings
warnings.filterwarnings("ignore")

```

In []:

```

#Load given dataset
data = p.read_csv('r_or_f.csv')
df=data.dropna()

```

In []:

```
df
```

In []:

```
df.columns
```

Splitting Train/Test:

In []:

```
#preprocessing, split test and dataset, split response variable  
X = df.drop(labels='fraudulent', axis=1)  
#Response variable  
y = df.loc[:, 'fraudulent']
```

In []:

```
#We'll use a test size of 30%. We also stratify the split on the response variable, which is very  
important to do because there are so few fraudulent transactions.  
from sklearn.model_selection import train_test_split  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)  
print("Number of training dataset: ", len(X_train))  
print("Number of test dataset: ", len(X_test))  
print("Total number of dataset: ", len(X_train)+len(X_test))
```

In []:

```
df.groupby('fraudulent').describe()
```

In []:

```
#plotting graph for distribution  
import matplotlib.pyplot as plt  
import seaborn as sns  
sns.countplot(x = "fraudulent", data = df)  
df.loc[:, 'fraudulent'].value_counts()  
plt.title('Distribution of fraudulent ')
```

In []:

```
df['fraudulent'].unique()  
Training model:
```

In []:

```
#!pip install nltk
```

In []:

```
import nltk
nltk.download('stopwords')
```

In []:

```
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
import re
import string

# remove whitespaces
df['description']=df['description'].str.strip()

# lowercase the text
df['description'] = df['description'].str.lower()

#remove punctuation
punc = string.punctuation
table = str.maketrans("",punc)
df['description']=df['description'].apply(lambda x: x.translate(table))

# tokenizing each message
df['word_tokens']=df.apply(lambda x: x['description'].split(' '),axis=1)

# removing stopwords
df['cleaned_text'] = df.apply(lambda x: [word for word in x['word_tokens'] if word not in
stopwords.words('english')],axis=1)

# stemming
ps = PorterStemmer()
df['stemmed']= df.apply(lambda x: [ps.stem(word) for word in x['cleaned_text']],axis=1)

# remove single letter words
df['final_text'] = df.apply(lambda x: ' '.join([word for word in x['stemmed'] if
len(word)>1]),axis=1)
```

In []:

```
# divide the set in training and test
from sklearn.model_selection import train_test_split
```

```
X,X_test,y,y_test = train_test_split(df.loc[:, 'description'],df['fraudulent'],test_size=0.2)
```

In []:

```
# Now we'll create a vocabulary for the training set with word count
```

```
from collections import defaultdict
```

```
vocab=defaultdict(int)
```

```
for text in X['final_text'].values:
```

```
    for elem in text.split(' '):
```

```
        vocab[elem]+=1
```

In []:

```
#!/pip install wordcloud
```

In []:

```
from wordcloud import WordCloud
```

```
real=' '.join(X.loc[y=='REAL','final_text'].values)
```

```
real_text = WordCloud(background_color='white',max_words=2000,width = 800, height =  
800).generate(real)
```

```
fake=' '.join(X.loc[y=='FAKE','final_text'].values)
```

```
fake_text = WordCloud(background_color='black',max_words=2000,width = 800, height =  
800).generate(fake)
```

```
plt.figure(figsize=[30,50])
```

```
plt.subplot(1,3,1)
```

```
plt.imshow(real_text,interpolation='bilinear')
```

```
plt.title("")
```

```
plt.axis('off')
```

```
plt.subplot(1,3,2)
plt.imshow(fake_text, interpolation='bilinear')
plt.axis('off')
plt.title("")
```

Module – 3

```
#import library packages
```

```
import pandas as p
```

```
import numpy as n
```

In []:

```
import warnings
```

```
warnings.filterwarnings("ignore")
```

In []:

```
#Load given dataset
```

```
data = p.read_csv('r_or_f.csv')
```

```
df=data.dropna()
```

In []:

```
df.columns
```

In []:

```
from sklearn.preprocessing import LabelEncoder
```

```
var_mod = ['Unnamed: 0', 'job_id', 'title', 'location', 'department',
           'salary_range', 'company_profile', 'description', 'requirements',
           'benefits', 'telecommuting', 'has_company_logo', 'has_questions',
           'employment_type', 'required_experience', 'required_education',
           'industry', 'function', 'fraudulent']
```

```
le = LabelEncoder()
```

```
for i in var_mod:
```

```
    df[i] = le.fit_transform(df[i]).astype(int)
```

In []:

#According to the cross-validated MCC scores, the random forest is the best-performing model, so now let's evaluate its performance on the test set.

```
from sklearn.metrics import confusion_matrix, classification_report, matthews_corrcoef,  
cohen_kappa_score, accuracy_score, average_precision_score, roc_auc_score
```

In []:

```
X = df.drop(labels='fraudulent', axis=1)
```

```
#Response variable
```

```
y = df.loc[:, 'fraudulent']
```

In []:

#We'll use a test size of 30%. We also stratify the split on the response variable, which is very important to do because there are so few fraudulent transactions.

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)
```

```
rfc = RandomForestClassifier()
```

```
rfc.fit(X_train, y_train)
```

```
predictR = rfc.predict(X_test)
```

```
print("")
```

```
x = (accuracy_score(y_test, predictR)*100)
```

```
print('Accuracy result of Random Forest Classifier is:', x)
```

```
print("")
```

```
print("")
```

```
print('Classification report of Random Forest Classifier : Results:')
```

```
print("")
```

```
print(classification_report(y_test, predictR))
```

```
xd = (accuracy_score(y_test,predictR)*100)
```

```
cm2=confusion_matrix(y_test,predictR)
print('Confusion Matrix result of Random Forest Classifier : is:\n',
confusion_matrix(y_test,predictR))
print("")
```

```
sensitivity1 = cm2[0,0]/(cm2[0,0]+cm2[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm2[1,1]/(cm2[1,0]+cm2[1,1])
print('Specificity : ', specificity1)
Logistic Regression:
```

In []:

```
from sklearn.linear_model import LogisticRegression
logR= LogisticRegression()
logR.fit(X_train,y_train)
predictR = logR.predict(X_test)
print("")
x = (accuracy_score(y_test,predictR)*100)
print('Accuracy result of Logistic Regression is:', x)
print("")

print("")
print('Classification report of Logistic Regression : Results:')
print("")

print(classification_report(y_test,predictR))
xl = (accuracy_score(y_test,predictR)*100)
```

```
cm2=confusion_matrix(y_test,predictR)
print('Confusion Matrix result of Logistic Regression : is:\n', confusion_matrix(y_test,predictR))
print("")
```

```
sensitivity1 = cm2[0,0]/(cm2[0,0]+cm2[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm2[1,1]/(cm2[1,0]+cm2[1,1])
print('Specificity : ', specificity1)
```

In []:

```
def graph():
    import matplotlib.pyplot as plt
    data=[xd,xl]
    alg="DT","LR"
    plt.figure(figsize=(10,5))
    b=plt.bar(alg,data,color=("green","yellow"))
    plt.title("Prediction of Opinion real or fake using ML",fontsize=15)
    plt.legend(b,data,fontsize=12)
    plt.savefig('comp.png')
```

In []:

```
graph()
```

In []:

```
import tkinter
from matplotlib.backends.backend_tkagg import (FigureCanvasTkAgg, NavigationToolbar2Tk)
from matplotlib.backend_bases import key_press_handler
from matplotlib.figure import Figure
import numpy as np
root = tkinter.Tk()
```

```

root.wm_title("Prediction of real or fake job using ML")
fig = Figure(figsize=(10,10),dpi=1)
canvas = FigureCanvasTkAgg(fig, master=root)
canvas.draw()
canvas.get_tk_widget().pack(side=tkinter.TOP, fill=tkinter.BOTH, expand=1)
icon=tkinter.PhotoImage(file='comp.png')
label=tkinter.Label(root,image=icon)
label.pack()
root.mainloop()

```

Module-4

```
import numpy as np
```

In []:

```
data=pd.read_csv("r_or_f.csv")
```

In []:

```
data.head()
```

In []:

```
data.shape
```

In []:

```
data.columns
```

In []:

```
data1=data.dropna()
```

```
data1.shape
```

In []:

```
data1.fraudulent.unique()
```

In []:

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

In []:

```
sns.countplot(x = "fraudulent", data=data1)
data1.loc[:, 'fraudulent'].value_counts()
plt.title('Distribution of fraudulent')
```

In []:

```
data1.fraudulent.value_counts()
```

```
import nltk
```

```
from nltk.corpus import stopwords
```

```
stopword=set(stopwords.words('english'))
```

```
stemmer = nltk.SnowballStemmer("english")
```

In []:

```
def clean(text):
```

```
    text = str(text).lower()
```

```
    text = re.sub('\[.*?\]', "", text)
```

```
    text = re.sub('https?://\S+|www\.\S+', "", text)
```

```
    text = re.sub('<.*?>+', "", text)
```

```
    text = re.sub('[%s]' % re.escape(string.punctuation), "", text)
```

```
    text = re.sub('\n', "", text)
```

```
    text = re.sub('\w*\d\w*', "", text)
```

```
    text = [word for word in text.split(' ') if word not in stopword]
```

```
    text="" ".join(text)
```

```
    text = [stemmer.stem(word) for word in text.split(' ')]
```

```
    text="" ".join(text)
```

```
    return text
```

```
data["description"] = data["description"].apply(clean)
```

In []:

```
x = np.array(data1["description"])
```

```
y = np.array(data1["fraudulent"])
```

In []:

```
x
```

```
In [ ]:
```

```
y
```

```
In [ ]:
```

```
from sklearn.feature_extraction.text import CountVectorizer  
from sklearn.model_selection import train_test_split
```

```
In [ ]:
```

```
cv = CountVectorizer()  
X = cv.fit_transform(x) # Fit the Data  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

```
In [ ]:
```

```
clf = LogisticRegression()  
clf.fit(X_train,y_train)
```

```
In [ ]:
```

```
pre=clf.predict(X_test)
```

```
In [ ]:
```

```
from sklearn.metrics import accuracy_score  
print(accuracy_score(y_test,pre)*100)
```

```
In [ ]:
```

```
input_word=input("ENTER THE SENTENCE:")
```

```
In [ ]:
```

```
data = cv.transform([input_word]).toarray()  
print(clf.predict(data))
```

Module-5

```
data=pd.read_csv("r_or_f.csv")
```

```
In [ ]:
```

```
data.head()
```

	In []:
<code>data.shape</code>	
	In []:
<code>data.columns</code>	
	In []:
<code>data1=data.dropna()</code>	
<code>data1.shape</code>	
	In []:
<code>print(data1.head())</code>	
	In []:
<code>data1.fraudulent.unique()</code>	
	In []:
<code>import matplotlib.pyplot as plt</code>	
<code>import seaborn as sns</code>	
	In []:
<code>sns.countplot(x = "fraudulent", data=data1)</code>	
<code>data1.loc[:, 'fraudulent'].value_counts()</code>	
<code>plt.title('Distribution of fraudulent')</code>	
	In []:
<code>data1.fraudulent.value_counts()</code>	
	In []:
<code>import string</code>	
<code>import nltk</code>	
<code>from nltk.corpus import stopwords</code>	
<code>stopword=set(stopwords.words('english'))</code>	
<code>stemmer = nltk.SnowballStemmer("english")</code>	
	In []:
<code>def clean(text):</code>	
<code>text = str(text).lower()</code>	

```

text = re.sub('\[.*?\]', '', text)
text = re.sub('https?://\S+|www\.\S+', '', text)
text = re.sub('<.*?>+', '', text)
text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
text = re.sub('\n', '', text)
text = re.sub('\w*\d\w*', '', text)
text = [word for word in text.split(' ') if word not in stopwords]
text="" ".join(text)
text = [stemmer.stem(word) for word in text.split(' ')]
text="" ".join(text)
return text
data["description"] = data["description"].apply(clean)

```

In []:

```

x = np.array(data1["description"])
y = np.array(data1["fraudulent"])

```

In []:

```

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split

```

In []:

```

cv = CountVectorizer()
X = cv.fit_transform(x) # Fit the Data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

```

In []:

```

from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier()
clf.fit(X_train,y_train)

```

In []:

```

pre=clf.predict(X_test)

```

In []:

```

from sklearn.metrics import accuracy_score

```

```
print(accuracy_score(y_test,pre)*100)
```

In []:

```
input_word=input("ENTER THE SENTENCE:")
```

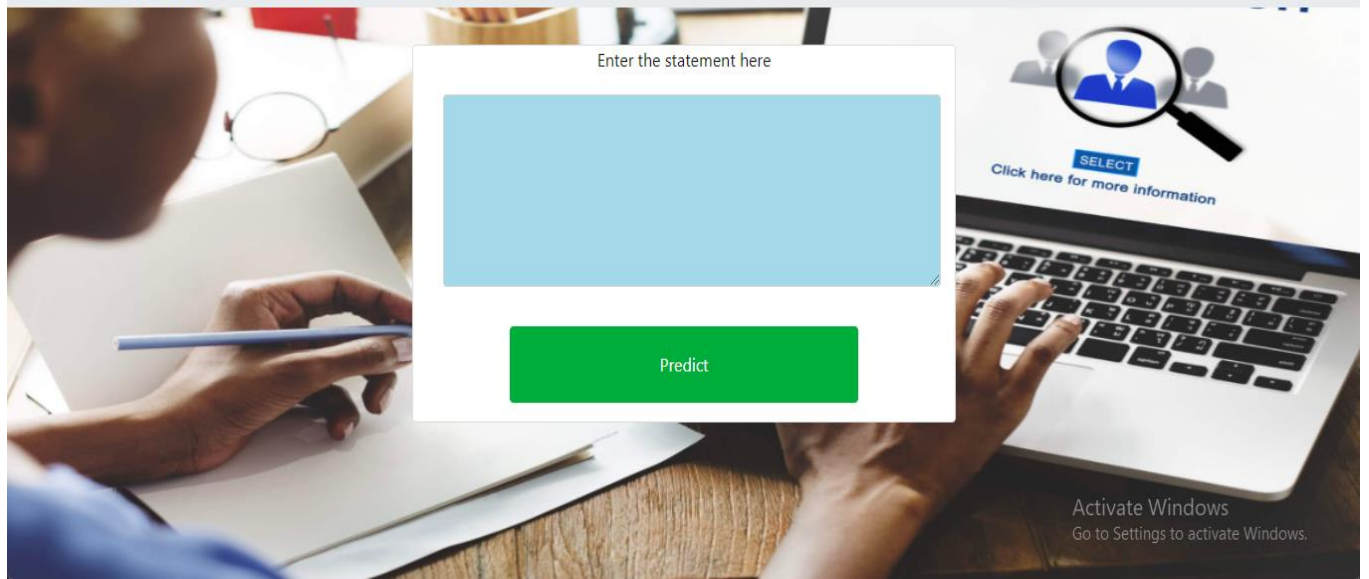
In []:

```
data = cv.transform([input_word]).toarray()  
print(clf.predict(data))
```

B.SCREENSHOTS

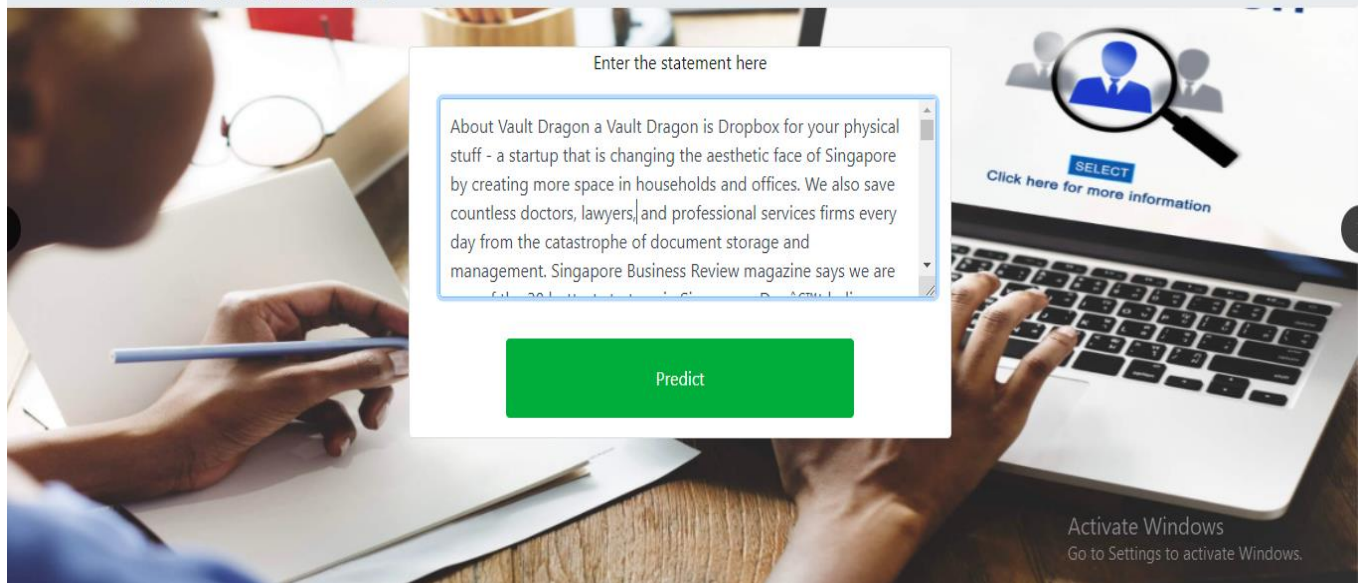
Machine Learning App with Flask

REAL OR FAKE JOB



Machine Learning App with Flask

REAL OR FAKE JOB



ML App

JOB PREDICTION



C.PUBLICATION WITH PLAGIARISM REPORT

ER24

by Green Official

Submission date: 17-Jan-2022 10:10AM (UTC-0800)
Submission ID: 1737164522
File name: CONFERENCE_PAPER-CSE123.pdf (351.26K)
Word count: 3571
Character count: 18467

REALAND FAKE JOB POSTING USING MACHINE LEARNING TECHNIQUE

KEERTHIGHA M.

Department of Computer Science and Engineering
Sathyabama Institute of Science and Technology
Chennai, India
keerthimaran2000@gmail.com

Mrs.REFONAA

Department of Computer Science and
Engineering, Chennai, India

V. ARTHI NANDHINI

Department of Computer Science and
Sathyabama Institute of Science and Technology
Chennai, India
arthislatspace@gmail.com

Abstract—Everything is being online now it allows people to reduce their manual efforts all of the job postings are posted online now so it gives the company a wide range of area to gather the talented candidates and for the people who are searching they can also know the information most companies can directly post the job. All job postings which are posted are not true there are fraudulent job postings also. So try to classify the fraudulent posting from real. The aim is to find the fraudulent job posting using Machine learning based concepts with correct accuracy. The analysis of dataset is done by Supervised ML to predict details like, variable finding, variate analysis, missed values treating and analysing data validation, cleaning data and visualizing the data would be carried on total dataset given.

rows are examine to view the dependent and independent variable and then various machine learnings techniques will be used to get the patterns and to get result with maximum accuries.

These reports are to the investigate of applicability of ml techniques for job posting classification. Atlast, it highspot few observation on the future findings, challenges, and their needs. The main idea is to do a model for original or fraud job Prediction finding result of matched accuracy by comparing them with supervised techniques.

Keywords — Fake Job, Online Recruitment, Machine Learning, Accuracy, Random forest algorithm.

I INTRODUCTION

The data driven will follow and store the represented data from a similar articles for fraud job detection. But these will still have many drawbacks: Because of the difficulties in gathering fraud jobs, existing dataset's capacity are low, and there are unverified jobs which misses data in similar articles which makes to identify the credibility of methods tough. Especially, difference between the fraud and true jobs are not limited to whether there is true feature in their similar article, but they include unseen difference at the linguistics level, such as the perspective of emotional expressions, writing flow. The method existing are hard to totally catch these difference. The proposed model has developed a machine learning model to find and split the original or fraud job posting to eradicate this method is to do a machine learning approach of GUI's user interface. The dataset is first preprocessed and the dataset is used to do the needful

II LITERATURE REVIEW

It is a form of data which tends to check the relevant points prevailing knowledge method wise approach to a significant theme. It is a another source of data and says published details in a particular topic and sometimes details in a specific topic in a span of time. Its main aim is to make the user to be with the recent knowledge with recent literature on data like hereafter the research which will use on places, follows presentation is a small collection files.

They provide an explanation of source / bind explanation and finds creative development in field, moreover it includes debate. On basis of the criteria , the review might estimate the files, tell user the most suitable / appropriate in these. It is a another source of data and says published details in a particular topic and sometimes details in a specific topic in a span of time.

Review of Literature Survey

Title : Predicting of Jobs Failures in Clouds Basis on Online Extreme Machine:

Author: CHUHONG LIU^{1, 2} , JINGJING HAN² , YNLEI SHANG¹ , CHANCHANG LIU¹ , BOJCHENG¹ , AND JINLIANG CHEN¹

Year: 2017

Early finding of job fails and basic disposal on previous can constantly increase the competence of resource intake on large number data centers. The prevailing machine learning-methods collectively get the doing ,working pattern, that cannot be taken for online guesses in practical data in which data arrive orderly.To eradicate these issues, a new model based on Online Extreme Learning Machine is presented in the paper to say and predict the online jobsterminations.

Title : Fraud Jobs Recruitment Finder Using Machine Learning Approach

Author: Samr Bandypadhyay, Shwni Duta

Year: 2020

To neglect fake post for jobs in the media, a tool which is automated using ML basis of methods are presented here. Various types are deployed for validating fake post for internet and final result for these are used in finding accurate scan finding model. This makes us easy for finding fraud work posting in a large value post collection. There are two main type of classifiers. However, experimental finding says that ensemble classifier are the accurate division to find scam on the single only classifier.

Title : Classification of job posting

Author: Ibrahim M. Nassef and Amjad H. Alzaann2

Year: 2020

In this, we found various ML classifiers which includes the following: Naive Bayes, Random Forest, Decision Tree algorithm, Support Vector Machine, KNN of data classified statement. The data that are been using here has original and fraud job posting. Applied TFIDF for creature extract and processed all the data. Hereafter, Evaluated all the data after implementing the classifiers. Evaluated metrics used are precisionate, recalling, f-measured, and accurate. For every classifier, all result are done and compared with each others.

Title : Online Recruitment fraud automatic detection

Author: Sokrats Vdros 1, Constantinos Kolias 2 ,

Year: 2017

The critical process of hiring has relatively recently been ported to the cloud. Especially, the automated systems that are responsible for finishing the recruitment process of new workers in an internet mode. The internet mode has a goal to make the hiring process more reliable and easy and accurate. But, the internet exposure of these business programs has proposed new procedures of failure which might cause some privacy losses for users and reduce organization reputation. So far, the employment scam is the most most common case of Online Recruitment Fraudulent. Unlike these online fraudulent problem the way of pullying of Online recruitment fraudulent has not yet git any good attention, remaining mostly unexplored till now.. At the same time, it lends and evaluates to ourself about the datasets that are available of 17,888 jobs adverts got from real life system.

Title : Enhanced RSA Algorithm using Fraud Modulus and Fraud Public Key Exponent

Author: Raghunathan K R, Ganesh Aithal, Suendra Shetty, Rakshih N

Year: 2018

In data communication , cryptography Public key cryptography has a most important role. There are two different keys in Public key to encrypt data and decrypt data The Public key cryptography has a efficient algorithm called RSA Efficiency of this is mostly dependent on how the public key components is shared that is modulus n and public key exponent e In this an enhanced RSA algorithm is presented to make high the complexity of the public keys paper have used unreal public key exponent r instead of e and Y instead of n . This Paper also uses standard metrics and gives comparative analysis of the proposed work .

III PROPOSED WORK

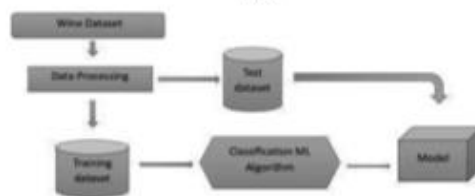


Figure 1. Architecture of Proposed model

To find whether a job post is fraud or not is the main target of this studies. By identifying and removing all these fraudulent job posts will make the job finders to concentrate on the original and real jobs. In this project a dataset is derived from Kaggle and it gives information about the jobs posted whether they belong to original or not original. The dataset has the theme as depicted..

job_id	int64
title	object
location	object
department	object
salary_range	object
company_profile	object
description	object
requirements	object
benefits	object
telecommuting	int64
has_company_logo	int64
has_questions	int64
employment_type	object
required_experience	object
required_education	object
industry	object
function	object
fraudulent	int64

Fig. 2. Schema of dataset

There are 17,888 job posts in this dataset above. The overall performance of this proposed methods are tested using this dataset. Some preprocessing algorithms are being used in this dataset before using these data to any classifiers. Pre- processing techniques include many algorithmic techniques. This makes the dataset to be transformed to a encoder to make them a vector. Several classifiers are used in these vectors. The following Fig. 3 says a description about the working criteria of a classifier for prediction.



Fig. 3.Detailed description for working of Classifiers

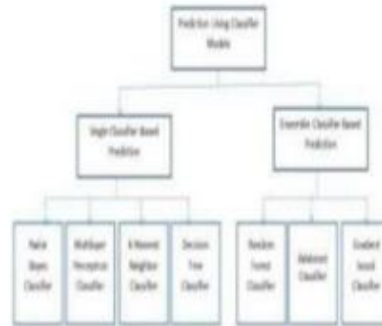


Fig. 4. Classification models used in this framework

As detailed in Fig. 4, a couple of classifiers are employed such as NB Classifier, Decision Tree Classifier, KNN Classifier, and Random Tree for classifying job post as fake. It is to be stated that the fake posts of the dataset is the main target of classification proposed. 85% of dataset is trained and again 15% of the total dataset is used for prediction process. The metrics like accuracy, F measurement, and CK score is used in finding and predicting classifiers. Atlast the classifier which has the best performance with all the metrics is given as the best candidate.

A. Implementing Classifiers

Classifiers here are being prepared with accurate parameter. The parameters that are default are not enough for increasing the performance of the models. The Best one for finding and eradicating the fraud jobs from job finders are made by adjusting the parameters which infact increases the reliability of the model.

MLP is used as a form of five concealed layers which include : 128,64,32,16,8 sizes respectively. Considering all the metrics The K- NN classifier predict a result, k as 5. On another fact, ensemble classifiers, like RF, Adaboost, GB classifiers are terminated in which fivehundred number result are built. Training data are allocated into this after proposing all the classification. Dataset is used in finding the result. Based on the result value and the original value the performance and activity are evaluated after the prediction is completed using the original and the original values.

B. Performance Evaluation Metrics

It is necessary to have some metrics in order to evaluate some skills of a model. For evaluating model accuracy is not enough accomplishment because it doesn't foretell correct cases.

This creates a problem if a fraudulent post is considered as a genuine post. So, it is important that the probabilities of the desirable and the non-desirable outputs is taken into account. To calculate this exactness it is required to be taken into account. Exactness tells us if the probability of the number of correct results over the number of correct obtained results. Recall gives number of correct result/ number of sample. F1-result is a framework is calculated as accuracy and recall. CK result is taken into account to calculate grade. Here the grade is a measurement which that gives the agreement for problem.

MS Error is used for the measurement of the various values between the predicted and the obtained result of test specimen. The values of all the MSE and precision, the F measure, CP score gives the better model.

WORKING



This proposed method is built a machine learning model to classify the real or fake job posting, a ML method is used in this model. The dataset is first preprocessed and the columns are analyzed to see the dependent and independent variable and various ML algorithms is implemented for extracting the desirable pattern to get the result with higher precision. Many data file from various location is merged to get a common data file, and various ML algorithm will be implemented to obtain

the desirable patterns to get the result with higher precision. Next, it loads, cleans and trims the data source for estimation. It must be taken into account that the steps must be followed carefully to clean. Data file gathered to predict data to Train and Test the file. Usually, 7:4 ratio is deployed to separate the Training Test file. Data structure that were formed with ML techniques is made to fall in line training file in accordance with the result precision and prediction is carried out at ease.

IV RESULT

The machine learning approach is developed in this model to predict the real or fake job posting. To view the dependent and independent variable it is preprocessed and examined, and to get the results various machine learning algorithms are used. Dataset are combined to form a common dataset and these would be used to get the pattern and result with higher accuracy. The data will be loaded, cleaned and trimmed for result.



V CONCLUSION AND FUTURE WORK

This project will be helpful to avoid the fake job notification to the job seekers to avoid any scam of any sort. This model is built starting from the process of dataset cleaning and preprocessing. This model will be used to find the fraudulent occurring during online hiring process. The Future work includes fraudulent job notification to connect with cloud. To amend work in AI domain.

REFERENCES

- [1] I. Rish, —An Empirical Study of the Naïve Bayes Classifier An empirical study of the naive Bayes classifier, *J no. January* 2001, pp. 41–46, 2014.
- [2] D. E. Walters, —Bayes's Theorem and the Analysis of Binomial Random Variables, *J Biometrical J.*, vol. 30, no. 7, pp. 817–825, 1988, doi: 10.1002/bimj.4710300710.
- [3] . Murtagh, —Multilayer perceptrons for classification and regression, *J Neurocomputing*, vol. 2, no. 5–6, pp. 183–197, 1991, doi: 10.1016/0925-2312(91)90023-5.
- [4] P. Cunningham and S. J. Delany, —K -Nearest Neighbour Classifiers, *J Mult. Classif. Syst.*, no. May, pp. 1–17, 2007, doi: 10.1016/S0031-3203(00)00099-6.
- [6] H. Sharma and S. Kumar, —A Survey on Decision Tree Algorithms of Classification in Data Mining, *J Int. J. Sci. Res.*, vol. 5, no. 4, pp. 2094–2097, 2016, doi: 10.21275/v5i4.nov162954.
- [7] E. G. Dada, J. S. Bassi, H. Chiroma, S. M. Abdulhamid, A. O. Adetunmbi, and O. E. Ajibuwa, "Machine learning for email spam filtering: review, approaches and open research problems, *J Heliyon*, vol. 5, no. 6, 2019, doi: 10.1016/j.heliyon.2019.e01802.
- [8] L. Breiman, —ST4 Method Random Forest, *J Mach Learn.*, vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1017/CBO9781107415324.004.
- [9] B. Biggio, I. Corona, G. Fumera, G. Giacinto, and a. F. Roli, —Bagging classifiers for fighting poisoning attacks in adversarial classification tasks, *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6713 LNCS, pp. 350–359, 2011, doi: 10.1007/978-3-642-21557-5_37.
- [10] A. Natekin and A. Knoll, —Gradient boosting machines, a tutorial, *J Front. Neurobot.*, vol. 7, no. DEC, 2013, doi: 10.3389/fnbot.2013.00021.
- [11] N. Hussain, H. T. Mirza, G. Rasool, I. Hussain, and a. M. Kaleem, —Spam review detection techniques: A systematic literature review, *J Appl. Sci.*, vol. 9, no. 5, pp. 1–26, 2019, doi: 10.3390/app9050987.
- [12] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu, a. —Fake News Detection on Social Media, *J ACM SIGKDD Explor. Newsl.*, vol. 19, no. 1, pp. 22–36, 2017, doi: 10.1145/3137597.3137600.
- [13] Shivam Bansal (2020, February). [Real or Fake] Fake Job Posting Prediction, Version 1. Retrieved March 29, 2020 from <https://www.kaggle.com/shivamb/real-or-fake-fake-job-posting-prediction>
- [14] H. M and S. M.N, —A Review on Evaluation Metrics for Data Classification Evaluations, *J Int. J. Data Min. Knowl. Manag. Process*, vol. 5, no. 2, pp. 01–11, 2015, doi:10.5121/ijdkp.2015.5201.
- [15] I. Rish, —An Empirical Study of the Naïve Bayes Classifier An empirical study of the naive Bayes classifier, *J no. January* 2001, pp. 41–46, 2014.
- [16] D. E. Walters, —Bayes's Theorem and the Analysis of Binomial Random Variables, *J Biometrical J.*, vol. 30, no. 7, pp. 817–825, 1988, doi: 10.1002/bimj.4710300710.
- [17] . Murtagh, —Multilayer perceptrons for classification and regression, *J Neurocomputing*, vol. 2, no. 5–6, pp. 183–197, 1991, doi: 10.1016/0925-2312(91)90023-5.
- [18] P. Cunningham and S. J. Delany, —K -Nearest Neighbour Classifiers, *J Mult. Classif. Syst.*, no. May, pp. 1–17, 2007, doi: 10.1016/S0031-3203(00)00099-6.
- [21] H. Sharma and S. Kumar, —A Survey on Decision Tree Algorithms of Classification in Data Mining, *J Int. J. Sci. Res.*, vol. 5, no. 4, pp. 2094–2097, 2016, doi: 10.21275/v5i4.nov162954.
- [22] E. G. Dada, J. S. Bassi, H. Chiroma, S. M. Abdulhamid, i. O. Adetunmbi, and O. E. Ajibuwa, "Machine learning for email spam filtering: review, approaches and open research problems, *J Heliyon*, vol. 5, no. 6, 2019, doi: 10.1016/j.heliyon.2019.e01802.
- [23] L. Breiman, —ST4 Method Random Forest, *J Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1017/CBO9781107415324.004.
- [24] B. Biggio, I. Corona, G. Fumera, G. Giacinto, and F. Roli, —Bagging classifiers for fighting poisoning attacks in adversarial classification tasks, *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6713 LNCS, pp. 350–359, 2011, doi: 10.1007/978-3-642-21557-5_37.
- [26] A. Natekin and A. Knoll, —Gradient boosting machines, a tutorial, *J Front. Neurobot.*, vol. 7, no. DEC, 2013, doi: 10.3389/fnbot.2013.00021.
- [27] N. Hussain, H. T. Mirza, G. Rasool, I. Hussain, and [28] M. Kaleem, —Spam review detection techniques: A systematic literature review, *J Appl. Sci.*, vol. 9, no. 5, pp. 1–26, 2019, doi: 10.3390/app9050987.
- [29] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu, —Fake News Detection on Social Media, *J ACM SIGKDD Explor. Newsl.*, vol. 19, no. 1, pp. 22–36, 2017, doi: 10.1145/3137597.3137600.
- [30] Shivam Bansal (2020, February). [Real or Fake] Fake Job Posting Prediction, Version 1. Retrieved March 29, 2020 from <https://www.kaggle.com/shivamb/real-or-fake-fake-job-posting-prediction>
- [31] H. M and S. M.N, —A Review on Evaluation Metrics for Data Classification Evaluations, *J Int. J. Data Min. Knowl. Manag. Process*, vol. 5, no. 2,

- [32] D. E. Walters, —Bayes's Theorem and the Analysis of Binomial Random Variables, *J Biometrical J.*, vol. 30, no. 7, pp. 817–825, 1988, doi: 10.1002/bimj.4710300710.
- [33] Murtagh, —Multilayer perceptrons for classification and regression, *J Neurocomputing*, vol. 2, no. 5–6, pp. 183–197, 1991, doi: 10.1016/0925-2312(91)90023-5.
- [34] P. Cunningham and S. J. Delany, —K-Nearest Neighbour Classifiers, *J Mult. Classif. Syst.*, no. May, pp. 1–17, 2007, doi: 10.1016/S0031-3203(00)00099-6.
- [35] H. Sharma and S. Kumar, —A Survey on Decision Tree Algorithms of Classification in Data Mining, *J Int. J. Sci. Res.*, vol. 5, no. 4, pp. 2094–2097, 2016, doi: 10.21275/v5i4.nov162954.
- [36] E. G. Dada, J. S. Bassi, H. Chiroma, S. M. Abdulhamid, A. O. Adetunmbi, and O. E. Ajibuwa, "Machine learning for email spam filtering: review, approaches and open research problems, *J Heliyon*, vol. 5, no. 6, 2019, doi: 10.1016/j.heliyon.2019.e01802.
- [37] L. Breiman, —ST4_Method_Random_Forest, *J Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1017/CBO9781107415324.004.
- [38] B. Biggio, I. Corona, G. Fumera, G. Giacinto, and F. Roli, —Bagging classifiers for fighting poisoning attacks in adversarial classification tasks," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6713 LNCS, pp. 350–359, 2011, doi: 10.1007/978-3-642-21557-5_37.
- [39] A. Natekin and A. Knoll, —Gradient boosting machines, a tutorial, *J Front. Neurobot.*, vol. 7, no. DEC, 2013, doi: 10.3389/fnbot.2013.00021.
- [40] N. Hussain, H. T. Mirza, G. Rasool, I. Hussain, and M. Kaleem, —Spam review detection techniques: A systematic literature review, *J Appl. Sci.*, vol. 9, no. 5, pp. 1–26, 2019, doi: 10.3390/app9050987.
- [41] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu, —Fake News Detection on Social Media, *J ACM SIGKDD Explor. News.*, vol. 19, no. 1, pp. 22–36, 2017, doi: 10.1145/3137597.3137600.
- [42] Shivam Bansal (2020, February). [Real or Fake] Fake JobPosting Prediction, Version 1. Retrieved March 29, 2020 from <https://www.kaggle.com/shivamb/real-or-fake-fake-job-posting-prediction>
- [43] H. M and S. M.N., —A Review on Evaluation Metrics for Data Classification Evaluations, *J Int. J. Data Min. Knowl. Manag. Process.*, vol. 5, no. 2,
- [44] D. E. Walters, —Bayes's Theorem and the Analysis of Binomial Random Variables, *J Biometrical J.*, vol. 30, no. 7, pp. 817–825, 1988, doi: 10.1002/bimj.4710300710.
- [45] Murtagh, —Multilayer perceptrons for classification and regression, *J Neurocomputing*, vol. 2, no. 5–6, pp. 183–197, 1991, doi: 10.1016/0925-2312(91)90023-5.
- [46] P. Cunningham and S. J. Delany, —K-Nearest Neighbour Classifiers, *J Mult. Classif. Syst.*, no. May, pp. 1–17, 2007, doi: 10.1016/S0031-3203(00)00099-6.
- [47] H. Sharma and S. Kumar, —A Survey on Decision Tree Algorithms of Classification in Data Mining, *J Int. J. Sci. Res.*, vol. 5, no. 4, pp. 2094–2097, 2016, doi: 10.21275/v5i4.nov162954.
- [48] E. G. Dada, J. S. Bassi, H. Chiroma, S. M. Abdulhamid, A. O. Adetunmbi, and O. E. Ajibuwa, "Machine learning for email spam filtering: review, approaches and open research problems, *J Heliyon*, vol. 5, no. 6, 2019, doi: 10.1016/j.heliyon.2019.e01802.
- [49] L. Breiman, —ST4_Method_Random_Forest, *J Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1017/CBO9781107415324.004.
- [50] B. Biggio, I. Corona, G. Fumera, G. Giacinto, and F. Roli, —Bagging classifiers for fighting poisoning attacks in adversarial classification tasks," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6713 LNCS, pp. 350–359, 2011, doi: 10.1007/978-3-642-21557-5_37.
- [51] A. Natekin and A. Knoll, —Gradient boosting machines, a tutorial, *J Front. Neurobot.*, vol. 7, no. DEC, 2013, doi: 10.3389/fnbot.2013.00021.
- [52] N. Hussain, H. T. Mirza, G. Rasool, I. Hussain, and M. Kaleem, —Spam review detection techniques: A systematic literature review, *J Appl. Sci.*, vol. 9, no. 5, pp. 1–26, 2019, doi: 10.3390/app9050987.

ER24

ORIGINALITY REPORT

4%

SIMILARITY INDEX

2%

INTERNET SOURCES

2%

PUBLICATIONS

2%

STUDENT PAPERS

PRIMARY SOURCES

- | | | |
|----------|---|-----|
| 1 | V. Nisha Jenipher, S. Radhika. "SVM kernel Methods with Data Normalization for Lung Cancer Survivability Prediction Application", 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), 2021
<small>Publication</small> | 1% |
| <hr/> | | |
| 2 | Submitted to Gitam University
<small>Student Paper</small> | 1% |
| <hr/> | | |
| 3 | www.preprints.org
<small>Internet Source</small> | <1% |
| <hr/> | | |
| 4 | "Emerging Trends in Computing and Expert Technology", Springer Science and Business Media LLC, 2020
<small>Publication</small> | <1% |
| <hr/> | | |
| 5 | Shawni Dutta, Samir Kumar Bandyopadhyay. "Forecasting of Campus Placement for Students Using Ensemble Voting Classifier", Asian Journal of Research in Computer Science, 2020
<small>Publication</small> | <1% |

Exclude quotes On

Exclude matches Off

Exclude bibliography On

THE END .