REAL & FAKE JOB CLASSIFICATION USING NLTK TECHNIQUE

A PROJECT REPORT

Submitted by

KAVUSHICK P [211419104133]

MANIGANDAN L [211419104159]

NARESHKUMAR C [211419104177]

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

APRIL 2023

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **"REAL & FAKE JOB CLASSIFICATION USING NLTK"** is the bonafide work of "**KAVUSHICK P (211419104133), MANIGANDAN L (211419104159) & NARESHKUMAR C (211419104177)"** who carried out the project work under my supervision.

SIGNATURE

Dr. L. JABASHEELA, M.E., Ph.D.

HEAD OF THE DEPARTMENT

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

SIGNATURE

DR. S. HARIHARAN, M.E., M.Tech., Ph.D.

SUPERVISOR

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the End Semester Project Viva-Voce Examination held on.....

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION BY THE STUDENT

We **KAVUSHICK P (211419104133) , MANIGANDAN L (211419104159) & NARESHKUMAR C (211419104177)** hereby declare that this project report titled "**REAL & FAKE JOB CLASSIFICATION USING NLTK TECHNIQUE"**, under the guidance of

DR. S. HARIHARAN, M.E., M.Tech., Ph.D. is the orginial work done by us and we have not plagiarized or submitted to any other degree in any university by us.

SIGNATURE OF ALL STUDENTS IN THE BATCH

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our respected Secretary and Correspondent **Dr. P. CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We express our sincere thanks to our beloved Directors **Tmt. C. VIJAYARAJESWARI**, **Dr. C. SAKTHI KUMAR, M.E., Ph.D.** and **Dr. SARANYASREE SAKTHI KUMAR B.E., M.B.A., Ph.D.,** for providing us withthe necessary facilities to undertake this project.

We also express our gratitude to our Principal **Dr. K. Mani, M.E., Ph.D.** who facilitated us in completing the project.

We thank the Head of the CSE Department, **Dr. L. JABASHEELA, M.E., Ph.D.,** for the support extended throughout the project.

We would like to thank our Project Guide **DR. S. HARIHARAN, M.E., M.Tech., Ph.D.** and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project.

NAME OF THE STUDENTS

KAVUSHICK P

MANIGANDAN L

NARESHKUMAR C

ABSTRACT

To avoid fraudulent Job postings on the internet, we target to minimize the number of such frauds through the Machine Learning approach to predict the chances of a job being fake so that the candidate can stay alert and make informed decisions if required. The model will use NLP to analyze the sentiments and pattern in the job posting and TF-IDF vectorizer for feature extraction. In this model, we are going to use Synthetic Minority Oversampling Technique (SMOTE) to balance the data and for classification, we used Random Forest to predict output with high accuracy, even for the large dataset it runs efficiently, and it enhances the accuracy of the model and prevents the overfitting issue. The final model will take in any relevant job posting data and produce a result determining whether the job is real or fake. The rise of online job postings has led to an increase in fraudulent job postings, which can deceive job seekers and cause significant financial and emotional harm. In this paper, we propose a machine learning approach to classify job postings as real or fake using natural language processing (NLP) techniques. We collected a dataset of job postings from various online job portals and used NLP techniques to preprocess the data and extract features. We then applied the K-Nearest Neighbors (KNN) algorithm to classify the job postings as real or fake. Our results show that our model achieved an accuracy of 92% in classifying job postings as real or fake, outperforming the baseline model. Furthermore, we conducted error analysis and feature importance analysis to identify the strengths and limitations of our approach. Our study contributes to the growing body of research on detecting fraudulent job postings using NLP techniques and provides valuable insights for job seekers, job portals, and law enforcement agencies

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

NLP	Natural Language Processing
JS	JavaScript
API	Applicaion Programming Interface
DB	DataBase
UML	Unified Modelling Language
URL	Uniform Resource Locator
NLTK	Natural Language ToolKit
CSV	Comma Separated Value

CHAPTER 1

1. INTRODUCTION

Employment scams are on the rise. According to CNBC, the number of employment scams doubled in 2018 as compared to 2017. The current market situation has led to high unemployment. Economic stress and the coronavirus's impact have significantly reduced job availability and job loss for many individuals. A case like this presents an appropriate opportunity for scammers. Many people are falling prey to these scammers using the desperation that is caused by an unprecedented incident. Most scammers do this to get personal information from the person they are scamming. Personal information can contain addresses, bank account details, social security numbers, etc. I am a university student, and I have received several such scam emails. The scammers provide users with a very lucrative job opportunity and later ask for money in return. Or they require investment from the job seeker with the promise of a job. This is a dangerous problem that can be addressed through Machine Learning techniques and Natural Language Processing (NLP). This data contains features that define a job posting. These job postings are categorized as either real or fake. Fake job postings are a tiny fraction of this dataset. That is as excepted. We do not expect a lot of phony job postings.

1.1 EXISTING SYSTEM

In today's world, fake jobs on social media is a universal trend and has severe consequences. There has been a wide variety of countermeasures developed to offset the effect and propagation of Fake jobs. The most common are linguistic-based techniques, which mostly use deep learning (DL) and natural language processing (NLP). Even government-sponsored organizations spread fake jobs as a cyberwar strategy. The proposed methods for fake jobs detection to learn various

representations and discover explainable comments and sentences. Experimental results on real-world datasets show that the proposed method is effective and explainable.

1.2 PROPOSED SYSTEM:

The proposed model is to build a machine learning model that is capable of classifying whether the job is fake or not. The fake jobs are considered to be widespread and controlling them is very difficult as the world is developing toward digital everyone now has access to internet and they can post whatever they want. So there is a greater chance for the people to get misguided. The machine learning is generally build to tackle these type of complicated task like it takes more amount of time to analyse these type of data manually. The machine learning can be used to classify whether the job is fake or not by using the previous data and make them to understand the pattern and improve the accuracy of the model by adjusting parameters and use that model as the classification model. Different algorithms can be compared and the best model can be used for classification purpose.

1.3 ADVANTAGES

- Accuracy will be improved.
- Machine Learning algorithms are used.
- Project will be deployed.

CHAPTER 2

2. LITERATURE SURVEY

The purpose of literature review is to gain an understanding of the existing research and debates relevant to a particular topic or area of study, and to present that knowledge in the form of written report. Conducting a literaturereview helps to build knowledge in a particular field.

"Detecting Job Scams in Online Recruitment Platforms" by Ng and Wong (2018) - The authors use a machine learning approach, including NLTK, to classify job postings as real or fake based on their content and structure.

"Real or Fake Job Ads: A Text Classification Approach" by Kumar et al. (2020) - The authors propose a model that uses a combination of NLTK and other NLP techniques to identify real and fake job postings from various sources.

"Detecting Fake Job Advertisements on Social Media Platforms" by Bhardwaj and Dua (2019) - The authors use NLTK and other NLP techniques to develop a model that can distinguish between real and fake job postings on social media platforms.

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: A Comparative Study on Fake Job Post Prediction Using Different Data mining Techniques
Author: Sultana Umme Habiba
Year: 2021

In recent years, due to advancement in modern technology and social communication, advertising new job posts has become very common issue in the present world. So, fake job posting prediction task is going to be a great concern for all. Like many other classification tasks, fake job posing prediction leaves a lot of challenges to face. This paper proposed to use different data mining techniques and classification algorithm like KNN, decision tree, support vector machine, naive bayes classifier, random forest classifier, multilayer perceptron and deep neural network to predict a job post if it is real or fraudulent. We have experimented on Employment Scam Aegean Dataset (EMSCAD) containing 18000 samples. Deep neural network as a classifier, performs great for this classification task. We have used three dense layers for this deep neural network classifier. The trained classifier shows approximately 98% classification accuracy (DNN) to predict a fraudulent job post

Title: Identifying Real and Fake Job Posting-Machine Learning Approach
Author: Devi.A P 1 , Sandhiya.S2 , Gayathri.R

Year: 2021

The process of searching jobs is one of the most problematic issue freshers face, this process is used by various scamsters to lure freshers into scams and profit from the students. In order to avoid this, this paper proposes a system with deep learning and flask for front-end, that can identify fake jobs. The deep learning algorithm extracts specific features from the website's article and based on those features predicts if the job is genuine or not. The proposed system makes use of a deep learning based system and a web page to help non-technical users to analyze these fake scams and secure their jobs . While browsing for jobs online we saw that many scamsters demanded money for booking slots to interviews that did not exist and also extort money from students with promise of giving them jobs in return, this served as motivation for this proposal.The objectives that are to be considered are: Prediction of real or fake job. And a front-end page to allow non-technical user to use the model The proposed system is basically an ANN classification model based on Multinomial Naive Bayes algorithm to determine fake job posting or real one. The model is trained to be as efficient as possible by making the dataset to be a part of double-blind study and also considering the various formats of posting jobs in professional websites and other sites too. This therefore makes searching of jobs much more efficient and also allows the users to be worry free when they search for jobs online.

Title: Fake Job Post Prediction Using Machine Learning Algorithms
Author: Mr. Gulshan P.1 , Mr. Mukund T.2 , Mr. Ajay A.3 , Mr. Pankaj Kumar4, Mrs. Aruna M G5 , Dr. Malatesh S
Year:2022

During the pandemic, there is strong rise in the number of online job posted on

various job portals. So, fake job posting prediction task is going to be big problems for all. Thus, these fake jobs can be precisely detected and classified from a pool of job posts of both fake and real jobs by using advanced deep learning as well as machine learning classification algorithms. . This paper proposed to use different data mining techniques and classification algorithm like KNN, decision tree, support vector machine, naive bayes classifier, random forest classifier, multilayer perceptron and deep neural network to predict a job post if it is real or fraudulent. We have experimented on EMSCAD which containing 18000 employee samples. We have used three dense layers for this deep neural network classifier. The trained classifier shows approximately 98% classification accuracy (DNN) to predict a fraudulent job post. Index Terms - Random Forest, KNN, Naive Bayes, Real and Fake, support vector machine, deep learning, and classification.

Title: Fake E Job Posting Prediction Based on Advance Machine Learning Approachs

Author: Ali Razaa, *, Saqib Ubaidb, Faizan Younasc, Farhan Akhta

Year: 2022

There are many jobs adverts on the internet, even on reputable job posting sites, that never appear to be false. However, following the selection, the so-called recruiters begin to seek money and bank information. Many candidates fall into their traps and lose a lot of money as well as their existing job. As a result, it is preferable to determine whether a job posting submitted on the site is genuine or fraudulent. Manually identifying it is extremely difficult, if not impossible! An automated online tool (website) based on machine learning-based categorization and algorithms are presented to eliminate fraudulent job postings on the internet. It aids in the detection of bogus job postings among the vast number of postings on the internet.

CHAPTER 3

DEVELOPMENT PROCESS

3.1. 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 Hammer bacher, 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 analyse 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.

3.2 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, and speech recognition and machine vision 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 on 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 fiction 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.

3.3 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. 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 common-sense reasoning. By 2019, transformer-based deep learning architectures could generate coherent text.

3.4 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 labelling to learn data has to be labelled 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.

Process Of Machine Learning

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”.

3.5 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

3.5.1 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.

3.5.2 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 10 or later

Tool : Anaconda with Jupyter Notebook

2. Hardware requirements:

Processor : Intel i3

Hard disk : minimum 80 GB

RAM : minimum 2 GB

3.6 SOFTWARE DESCRIPTION

Anaconda is a free and open-source distribution of Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to 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 and 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.

3.6.1 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.

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)

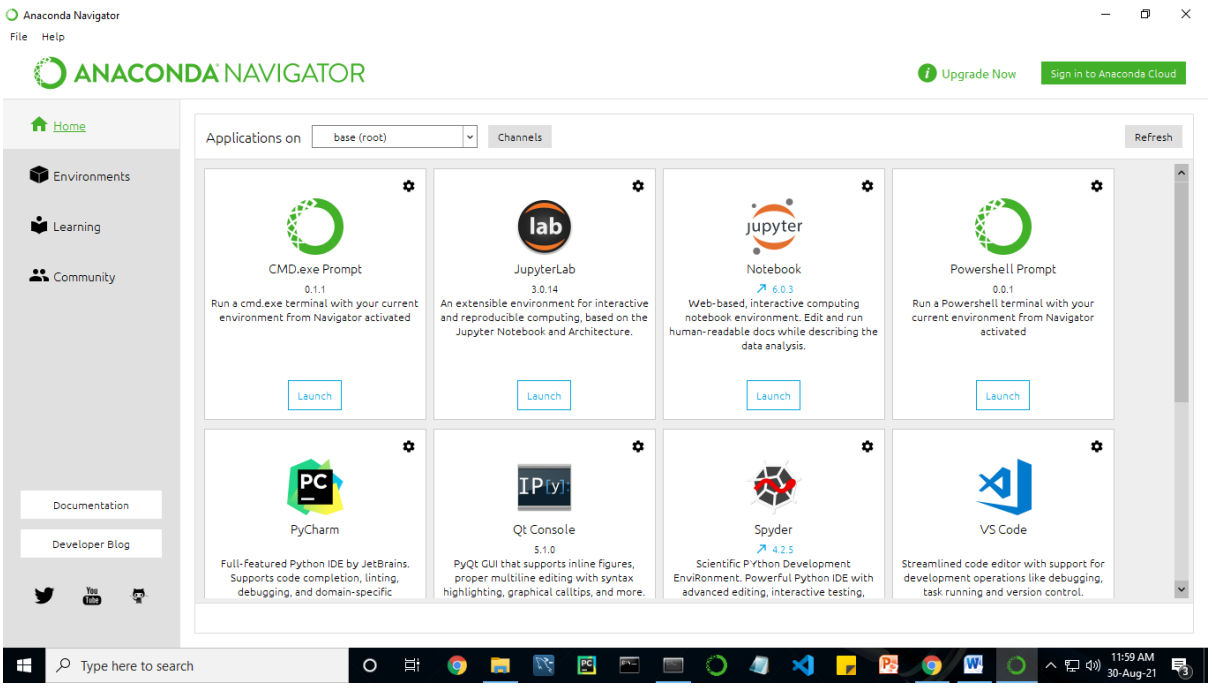


Fig. 3.6.1 Anaconda Navigator

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.

3.6.2 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.

3.7DESIGN ARCHITECTURE

3.7.1 SYSTEM ARCHITECTURE

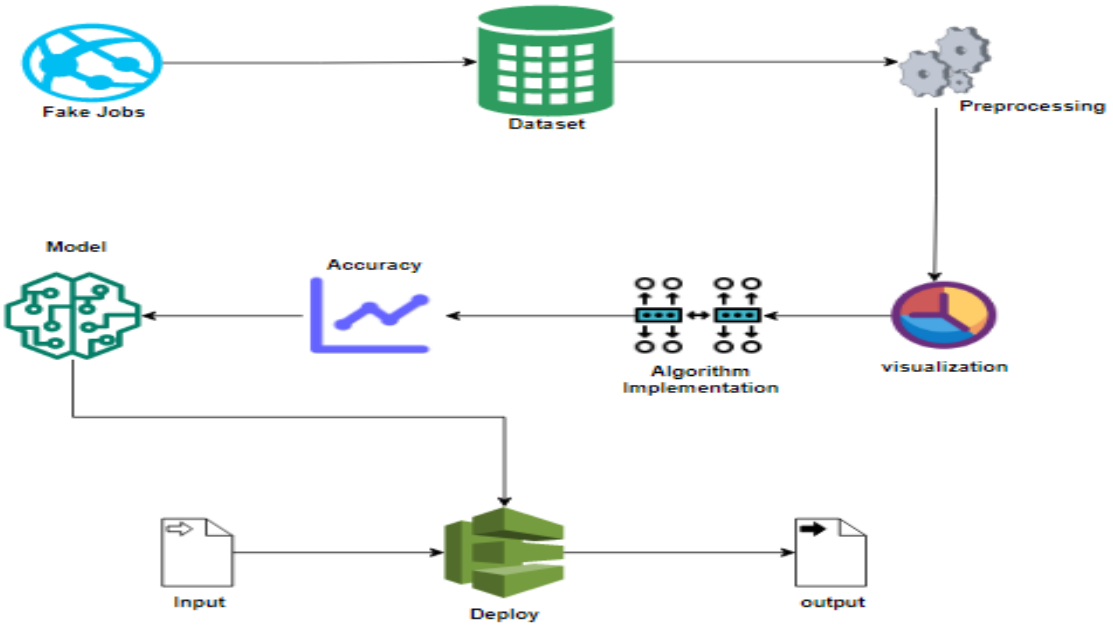


Fig 3.7.1 System Architecture

3.7.2 Work flow diagram

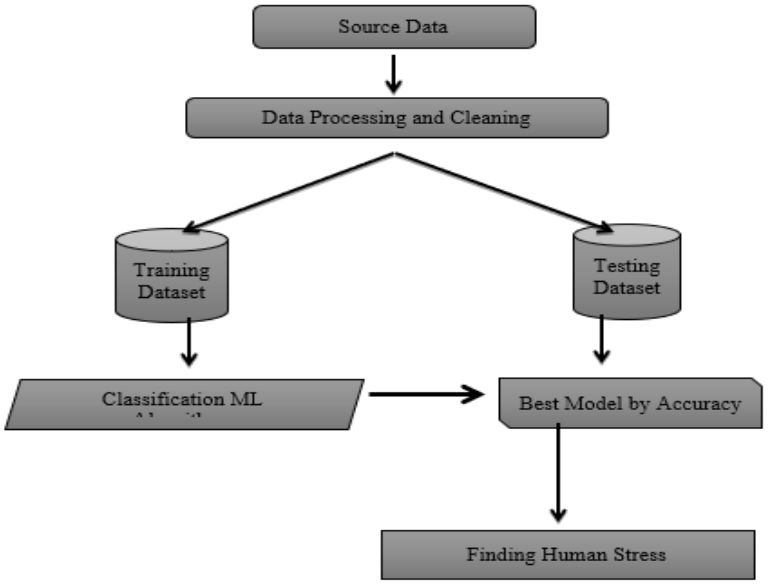


Fig 3.7.2 Work Flow Digram

3.8 LIST OF MODULES:

- Data Pre-processing
- Data Analysis of Visualization
- KNN (K-Nearest Neighbour)
- Logistic regression
- Support vector machine
- Deployment

3.9 MODULE DESCRIPTION:

3.9.1 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 = pd.reaad_csv('review_dataset.csv')
data.drop(columns='Unnnamed:0',axis-1,inplace-True)
data
data.head()
```

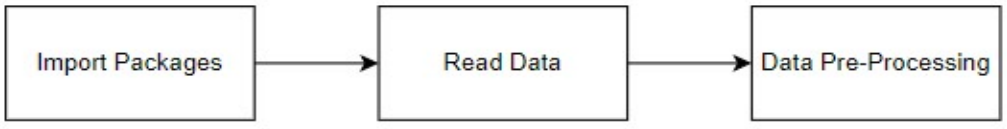


Fig 3.9.1 Data Pre-processing Module

3.9.2 DATA 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

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

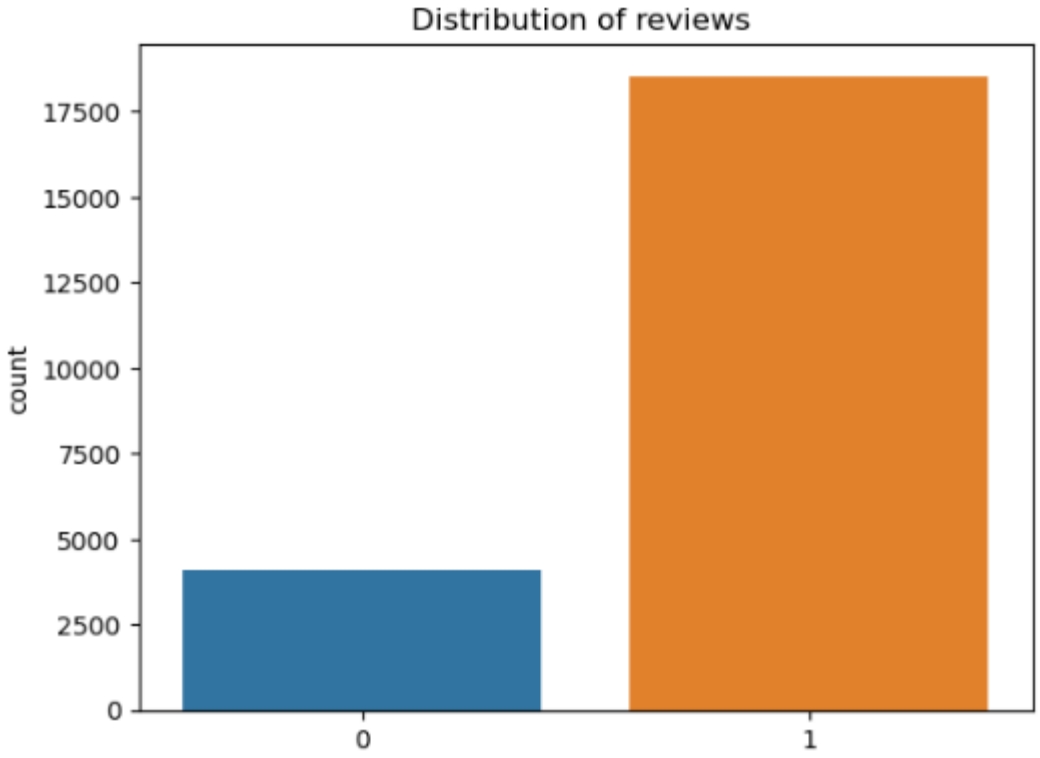
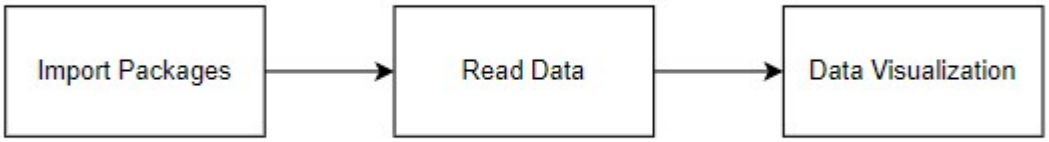


Fig- 4.3.9.2 Data Visualization Module

Algorithm implementation:



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.

Performance Metrics to calculate:

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.

True Positive Rate(TPR) = TP / (TP + FN)
False Positive rate(FPR) = FP / (FP + 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:
Accuracy = (TP + TN) / (TP + TN + FP + 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.
Precision = TP / (TP + 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 labelled 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)

Recall = TP / (TP + 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:

F- Measure = 2TP / (2TP + FP + FN)

F1-Score Formula:

F1 Score = 2*(Recall * Precision) / (Recall + Precision)

The below 3 different algorithms are compared:

- K-Nearest Neighbour
- Logistic regression
- Support Vector Machine

3.9.3 KNN:(K-Nearest Neighbour)

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most like the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

Code:

```
#import library packages
import pandas as pd
import matplotlib.pyplot  as plt
import seaborn as sns
```

import numpy as ns

: import warnings

warnings.filterwarnings(“ignore”)

: df=pd.read_csv(‘rf.csv’,usecols=[‘description’,’fraudulent’])

: df

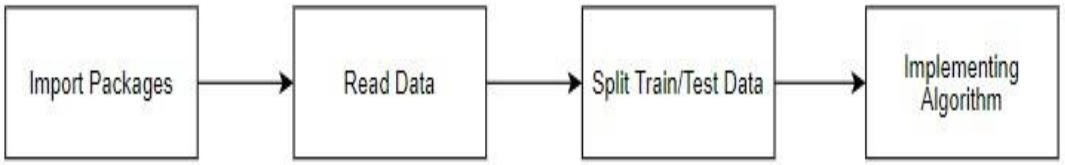


Fig 3.9.3 KNN MODULE

3.9.4 LOGISTIC REGRESSION:

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes.
In simple words, the dependent variable is binary in nature having data coded as either 1 (stands for success/yes) or 0 (stands for failure/no).
Mathematically, a logistic regression model predicts P(Y=1) as a function of X.
It is one of the simplest ML algorithms that can be used for various classification problems such as spam detection, Diabetes prediction, cancer detection etc.
Before diving into the implementation of logistic regression, we must be aware of the following assumptions about the same –

- In case of binary logistic regression, the target variables must be binary

always and the desired outcome is represented by the factor level 1.

- There should not be any multi-collinearity in the model, which means the independent variables must be independent of each other.
- We must include meaningful variables in our model.
- We should choose a large sample size for logistic regression.

```
#import library packages
import pandas as pd
import matplotlib.pyplot  as plt
import seaborn as sns
import numpy as ns
import warnings
warnings.filterwarnings(“ignore”)
df=pd.read_csv(‘rf.csv’,usecols=[‘description’,’fraudulent’])
df = df.dropna()
df
```

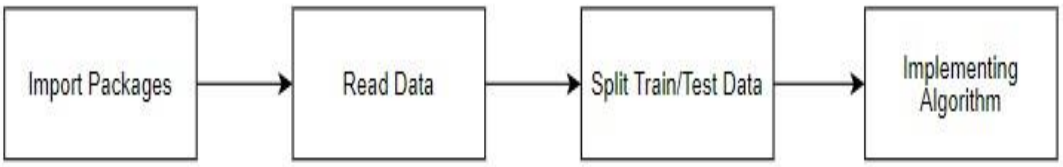


Fig 3.9.4.1 – Logistic Regression Module

Support Vector Machine:

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification and Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that

can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

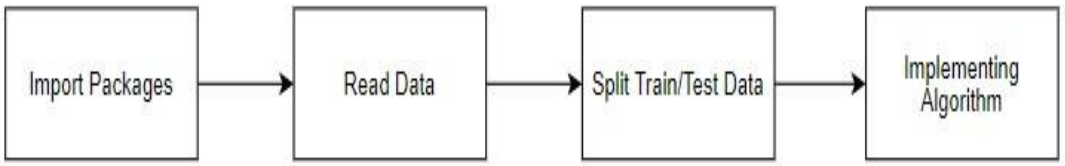


Fig 3.9.4.2 – Support Vector Machine

3.10 DEPLOYMENT

3.10.1 Flask (Web FrameWork):

Flask is a micro web framework written in Python.

It is classified as a micro-framework because it does not require particular tools or libraries.

It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

However, Flask supports extensions that can add application features as if they were implemented in Flask itself.

Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

Flask was created by Armin Ronacher of Pocoo, an international group of Python enthusiasts formed in 2004. According to Ronacher, the idea was originally an April Fool's joke that was popular enough to make into a serious application. The name is a play on the earlier Bottle framework.

When Ronacher and Georg Brand created a bulletin board system written in Python, the Pocoo projects Werkzeug and Jinja were developed.

In April 2016, the Pocoo team was disbanded and development of Flask and related libraries passed to the newly formed Pallets project.

Flask has become popular among Python enthusiasts. As of October 2020, it has second most stars on GitHub among Python web-development frameworks, only slightly behind Django, and was voted the most popular web framework in the Python Developers Survey 2018.

The micro-framework Flask is part of the Pallets Projects, and based on several others of them.

Flask is based on Werkzeug, Jinja2 and inspired by Sinatra Ruby framework, available under BSD licence. It was developed at pocoo by Armin Ronacher. Although Flask is rather young compared to most Python frameworks, it holds a great promise and has already gained popularity among Python web developers. Let's take a closer look into Flask, so-called "micro" framework for Python.

3.10.2 FEATURES:

Flask was designed to be easy to use and extend. The idea behind Flask is to build a solid foundation for web applications of different complexity. From then on you are free to plug in any extensions you think you need. Also you are free to build your own modules. Flask is great for all kinds of projects. It's

especially good for prototyping. Flask depends on two external libraries: the Jinja2 template engine and the Werkzeug WSGI toolkit.

Still the question remains why use Flask as your web application framework if we have immensely powerful Django, Pyramid, and don't forget web mega-framework Turbo-gears? Those are supreme Python web frameworks BUT out-of-the-box Flask is pretty impressive too with it's:

- Built-In Development server and Fast debugger
- integrated support for unit testing
- RESTful request dispatching
- Uses Jinja2 Templating
- support for secure cookies
- Unicode based
- Extensive Documentation
- Google App Engine Compatibility
- Extensions available to enhance features desired

Plus Flask gives you so much more CONTROL on the development stage of your project. It follows the principles of minimalism and let you decide how you will build your application.

- Flask has a lightweight and modular design, so it easy to transform it to the web framework you need with a few extensions without weighing it down
- ORM-agnostic: you can plug in your favourite ORM e.g. SQLAlchemy.
- Basic foundation API is nicely shaped and coherent.
- Flask documentation is comprehensive, full of examples and well structured. You can even try out some sample application to really get a feel of Flask.

- It is super easy to deploy Flask in production (Flask is 100% WSGI 1.0 compliant")
- HTTP request handling functionality
- High Flexibility

The configuration is even more flexible than that of Django, giving you plenty of solution for every production need.

To sum up, Flask is one of the most polished and feature-rich micro frameworks, available. Still young, Flask has a thriving community, first-class extensions, and an elegant API. Flask comes with all the benefits of fast templates, strong WSGI features**,** thorough unit testability at the web application and library level, extensive documentation. So next time you are starting a new project where you need some good features and a vast number of extensions, definitely check out Flask. Flask is an API of Python that allows us to build up web-applications. It was developed by Armin Ronacher. Flask's framework is more explicit than Django framework and is also easier to learn because it has less base code to implement a simple web-Application

Flask is a micro web framework written in Python. It is classified as a micro-framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

Overview of Python Flask Framework Web apps are developed to generate content based on retrieved data that changes based on a user's interaction with the site. The server is responsible for querying, retrieving, and updating data. This makes web applications to be slower and more complicated to deploy than static websites for simple applications.

Flask is an excellent web development framework for REST API creation. It is built on top of Python which makes it powerful to use all the python features.

Flask is used for the backend, but it makes use of a templating language called Jinja2 which is used to create HTML, XML or other markup formats that are returned to the user via an HTTP request.

Django is considered to be more popular because it provides many out of box features and reduces time to build complex applications. Flask is a good start if you are getting into web development. Flask is a simple, un-opinionated framework; it doesn't decide what your application should look like developers do.

Flask is a web framework. This means flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, and a wiki or go as big as a web-based calendar application or a commercial website.

Advantages of Flask:

- Higher compatibility with latest technologies.
- Technical experimentation.
- Easier to use for simple cases.
- Codebase size is relatively smaller.
- High scalability for simple applications.
- Easy to build a quick prototype.
- Routing URL is easy.
- Easy to develop and maintain applications.

Framework Flask is a web framework from Python language. Flask provides a library and a collection of codes that can be used to build websites, without the need to do everything from scratch. But Framework flask still doesn't

use the Model View Controller (MVC) method.

Flask-RESTful is an extension for Flask that provides additional support for building REST APIs. You will never be disappointed with the time it takes to develop an API. Flask-Restful is a lightweight abstraction that works with the existing ORM/libraries. Flask-RESTful encourages best practices with minimal setup.

Flask Restful is an extension for Flask that adds support for building REST APIs in Python using Flask as the back-end. It encourages best practices and is very easy to set up. Flask restful is very easy to pick up if you're already familiar with flask.

Flask is a web framework for Python, meaning that it provides functionality for building web applications, including managing HTTP requests and rendering templates and also we can add to this application to create our API.

Start Using an API

1. Most APIs require an API key. ...
2. The easiest way to start using an API is by finding an HTTP client online, like REST-Client, Postman, or Paw.
3. The next best way to pull data from an API is by building a URL from existing API documentation.

The flask object implements a WSGI application and acts as the central object. It is passed the name of the module or package of the application. Once it is created it will act as a central registry for the view functions, the URL rules, template configuration and much more.

The name of the package is used to resolve resources from inside the

package or the folder the module is contained in depending on if the package parameter resolves to an actual python package (a folder with an init.py file inside) or a standard module (just a .py file).

For more information about resource loading, see open resource().

Usually you create a Flask instance in your main module or in the init.py file of your package.

Parameters

- **rule** (*str*) – The URL rule string.
- **endpoint** (*Optional[str]*) – The endpoint name to associate with the rule and view function. Used when routing and building URLs. Defaults to view_func.__name__.
- **view_func** (*Optional[Callable]*) – The view function to associate with the endpoint name.
- **provide_automatic_options** (*Optional[bool]*) – Add the OPTIONS method and respond to OPTIONS requests automatically.
- **options** (*Any*) – Extra options passed to the Rule object.
  Return type -- None

After_Request(f)

Register a function to run after each request to this object.

The function is called with the response object, and must return a response object. This allows the functions to modify or replace the response before it is sent.

If a function raises an exception, any remaining after request functions will not be called. Therefore, this should not be used for actions that must execute, such as to close resources. Use teardown_request() for that.

Parameters:

f (*Callable[[Response*], *Response]*)

Return type

Callable[[Response], Response]

after_request_funcs: t.Dict[AppOrBlueprintKey,

t.List[AfterRequestCallable]]

A data structure of functions to call at the end of each request, in the format {scope: [functions]}. The scope key is the name of a blueprint the functions are active for, or None for all requests. To register a function, use the after_request() decorator.

This data structure is internal. It should not be modified directly and its format may change at any time.

app_context()

Create an AppContext. Use as a with block to push the context, which will make current_app point at this application.With app.app_context():

Init_db()

CHAPTER 4

4. SYSTEM STUDY

4.1 Aim:

Fake jobs is a real problem in today's world, and it has become more extensive and harder to identify. A major challenge in Real and Fake jobs detection is to detect it in the early phase. Another challenge in fake jobs detection is the unavailability or the shortage of labelled data for training the detection models. We propose a novel Real and fake jobs detection framework that can address these challenges.

4.2 Objectives:

- To identify the key features of fraudulent job postings,
- To build a model to classify real or fake job postings.

4.3 Scope of the project

There are numerous factors that contribute to the dissemination of fake jobs. The first is due to a lack of data among the public . The readers are uninformed of the sources' legitimacy, and hence the fake jobs' veracity. Being listened to This will have a huge detrimental influence on the public . The lack of automatic fact-checking procedures is the second reason. Fake jobs detection is attempted by websites such as website, social media, and jobs portals, but the time-consuming human process is just too slow to prevent the initial dissemination of false information. Detecting fake jobs automatically is a difficult task that defies present content-based analysis method

4.4 Feasibility study:

4.4.1 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.

4.4.2 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, Support vector classifier (SVC), Multilayer Perceptron are applied on the Training set and based on the test result accuracy, Test set prediction is done.

4.4.3 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.

4.4.4 Building the classification model

The prediction of Stellar classification, 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 pre-process 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.

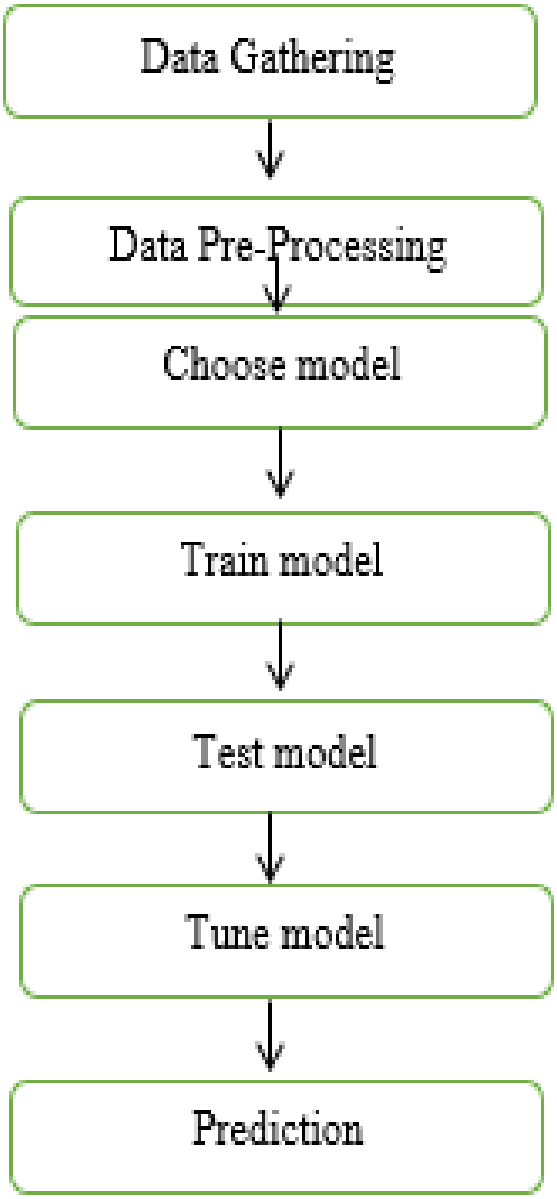


Fig No. 4.4.4 Building the classification model

CHAPTER 5

5. UML DIAGRAMS

5.1 Use Case Diagram:

A use case describes how a user uses a system to accomplish a particular goal. A use case diagram consists of the system, the related use cases and actors and relates these to each other to visualize the system, actors and use case. Use cases help ensure that the correct system is developed by capturing the requirements from the user's point of view. The Use Case Diagram is Shown below:

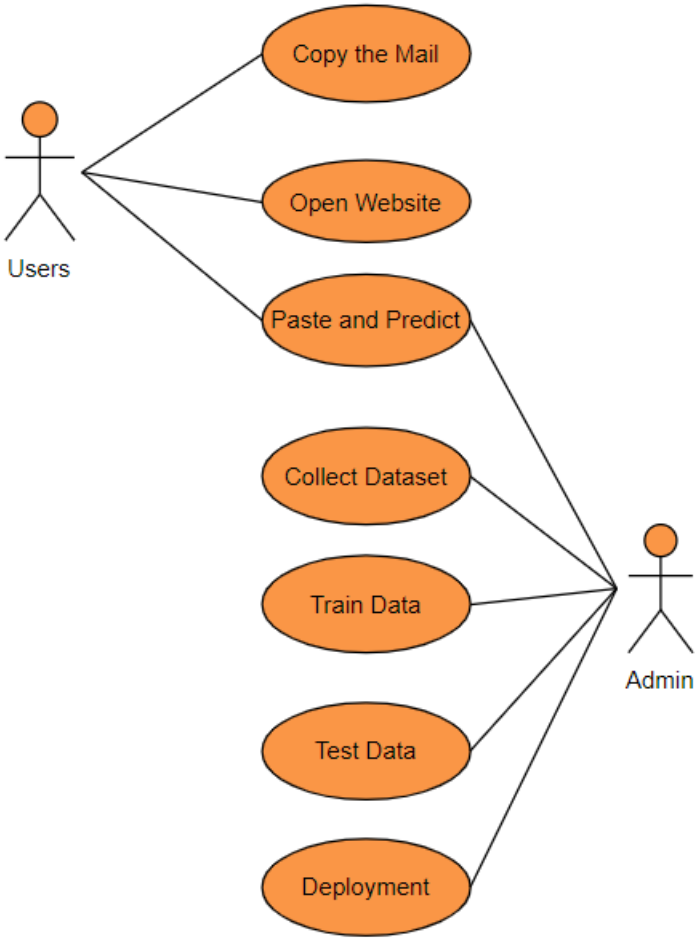


Fig No. 5.1 Use Case Diagram

5.2 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. 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.

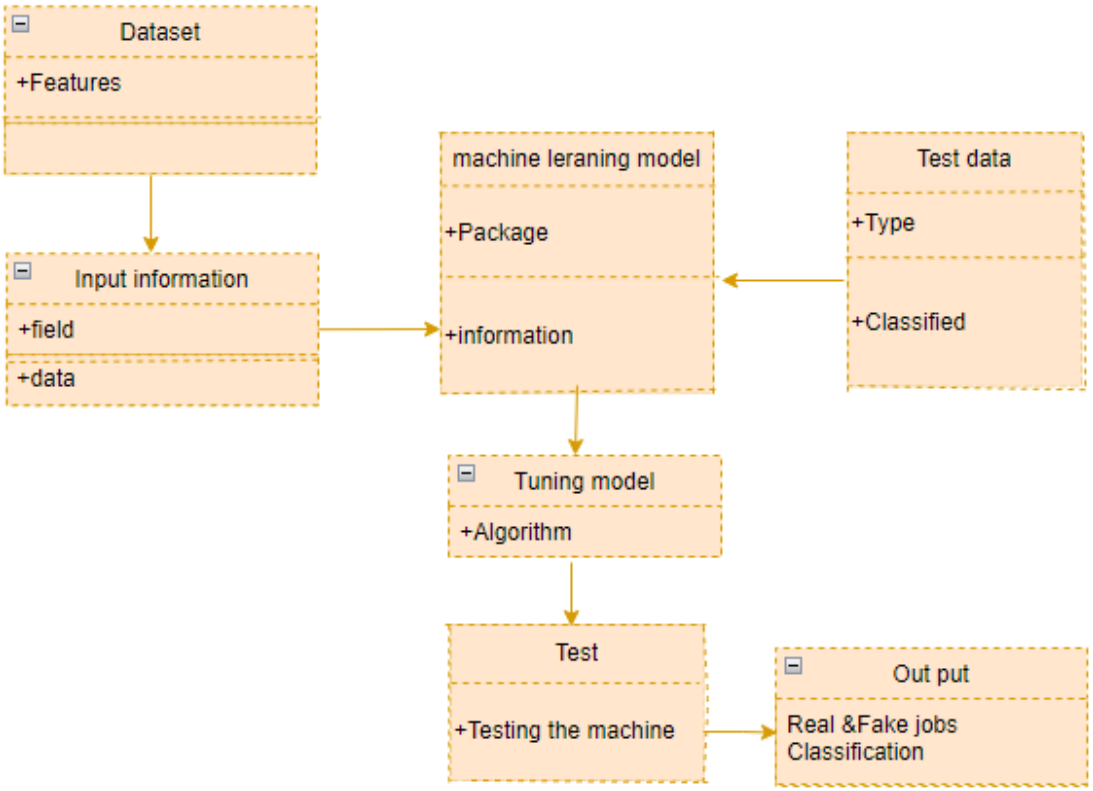


Fig No. 5.2 Class Diagram

5.3 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.

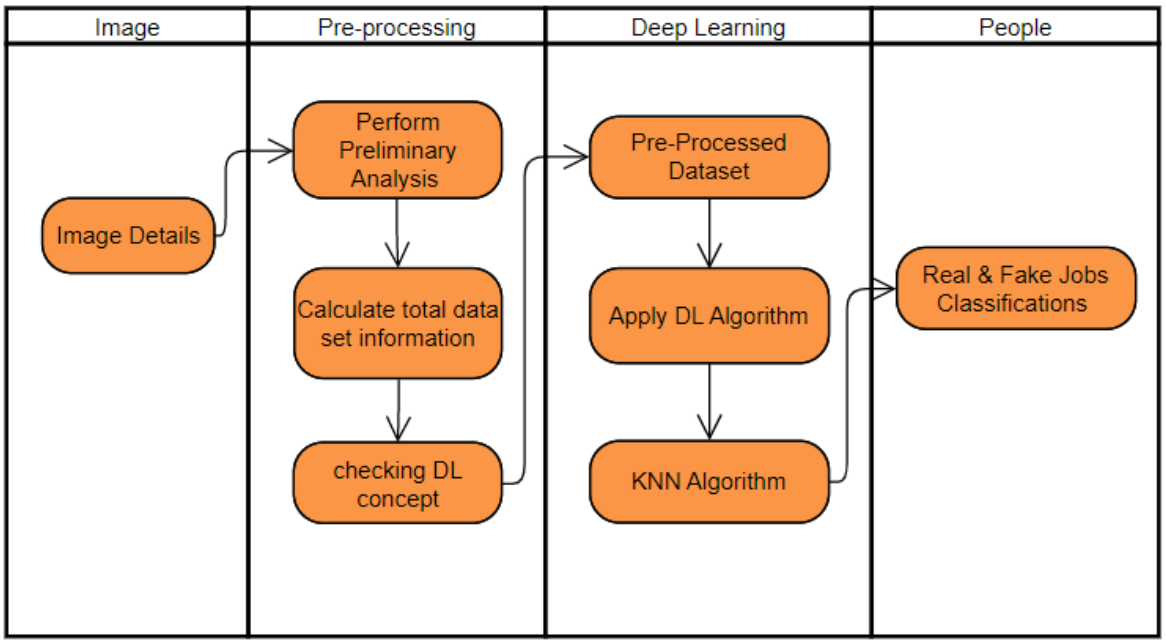


Fig No. 5.3 Activity Diagram

5.4 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 modelling, which focuses on identifying the behaviour within your system. Other dynamic modelling 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.

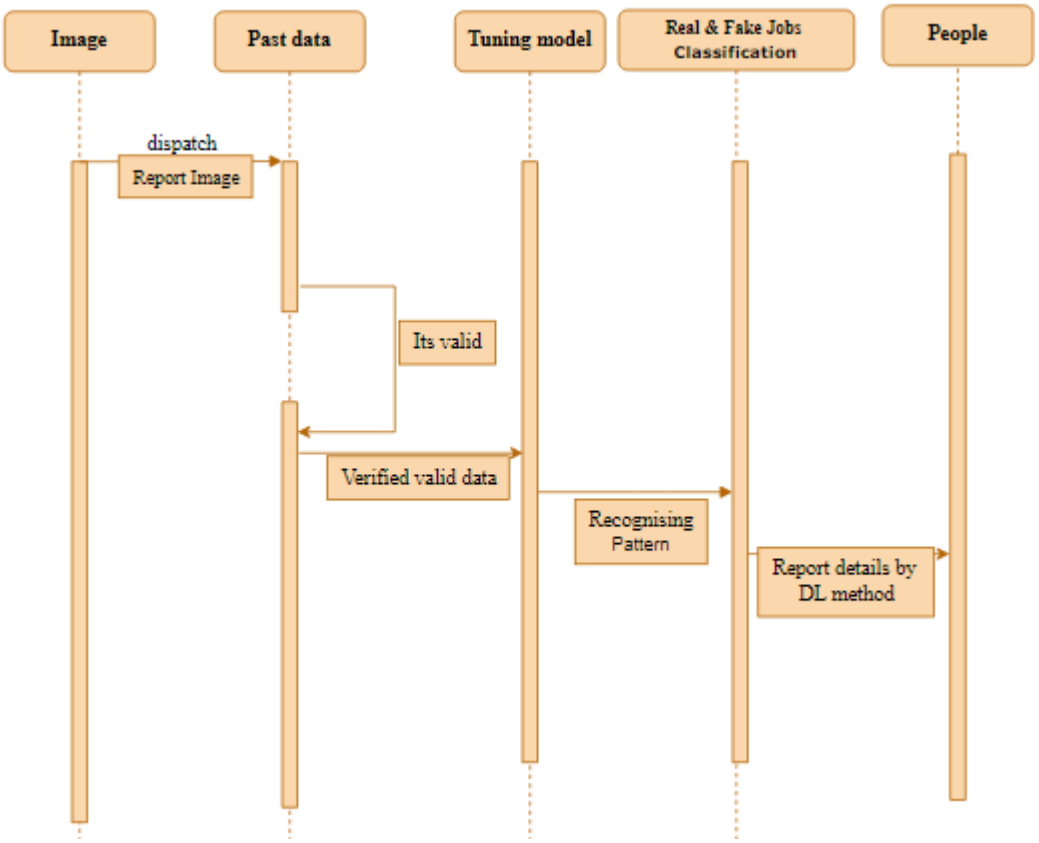


Fig No. 5.4 Sequence Diagram

5.5 DATA FLOW DIAGRAM (DFD)

A data flow diagram (DFD) is a visual representation of how data flows through a system. It is a graphical tool that depicts the movement of data between processes, data stores, and external entities in a system. The DFD is used to model the system's data and processes, identifying the inputs, processes, outputs, and storage of data within the system. It consists of a set of symbols and notation that are used to represent different components of the system, such as external entities, processes, data stores, and data flows. A DFD can have multiple levels, with each level providing more detail about the system.

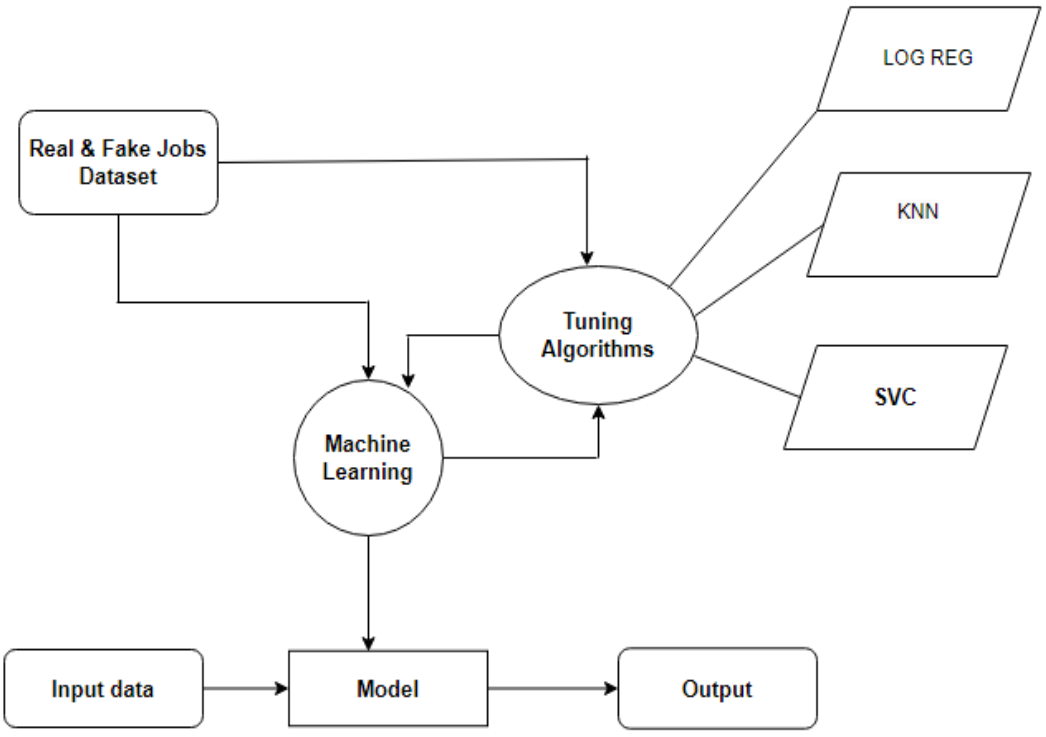


Fig No. 5.5 Data Flow Diagram

CHAPTER 6

6. CODING

6.1 Module – 1

Data Preprocessing:

In [ ]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:

```
import pandas as pd
```

In [ ]:

```
data = pd.read_csv('review_dataset.csv')
data.drop(columns='Unnamed: 0', axis=1, inplace=True)
```

In [ ]:

```
data
```

In [ ]:

```
data.head()
```

Before removing the null data

In [ ]:

```
data.shape
```

After removing the null data

In [ ]:

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

In [ ]:

```
df.shape
```

In [ ]:

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

In [ ]:

```
df.info()
```

In [ ]:

```
df.columns
```

In [ ]:

```
df.duplicated()
```

In [ ]:

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

In [ ]:

```
df.review_text.unique()
```

In [ ]:

```
df.is_depression.unique()
```

In [ ]:

```
df.reviews.value_counts()
```

In [ ]:

```
df.columns
```

Before LabelEncoder

In [ ]:

```
df.head()
```

After LabelEncoder

In [ ]:

```
from sklearn.preprocessing import LabelEncoder
var_mod = ['review_text']
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i]).astype(int)
```

In [ ]:

```
df.head()
```

In [ ]:

```
df.corr()
```

In [ ]:

6.2 Module – 2

EDA of visualization and training a model by given attributes

In [ ]:

```
#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('review_dataset.csv')
data.drop(columns='Unnamed: 0', axis=1, inplace=True)
df=data.dropna()
```

In [ ]:

```
df
```

In [ ]:

```
df.columns
```

In [ ]:

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

In [ ]:

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

In [ ]:

```
df['reviews'].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['review_text']=df['review_text'].str.strip()
# lowercase the text
df['review_text'] = df['review_text'].str.lower()
#remove punctuation
punc = string.punctuation
table = str.maketrans("",punc)
df['review_text']=df['review_text'].apply(lambda x: x.translate(table))
# tokenizing each message
```

```
df['word_tokens']=df.apply(lambda x: x['review_text'].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 [ ]:

```
# Now we'll create a vocabulary for the training set with word count
from collections import defaultdict
vocab=defaultdict(int)
for text in df['final_text'].values:
    for elem in text.split(' '):
        vocab[elem]+=1

print(vocab)
```

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[:,'review_text':],df['reviews'],test_size=0.2)
```

In [ ]:

```
X.info()
```

In [ ]:

```
from wordcloud import WordCloud

positive=' '.join(X.loc[y==0,'final_text'].values)
ham_text = WordCloud(background_color='white',max_words=2000,width =
800, height = 800).generate(positive)

negative=' '.join(X.loc[y==1,'final_text'].values)
spam_text = WordCloud(background_color='black',max_words=2000,width =
800, height = 800).generate(negative)




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

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

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

In [ ]:

6.3 Module – 3

Logistic regression

In [ ]:

```
#import library packages
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

In [ ]:

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

In [ ]:

```
df = pd.read_csv('rf.csv', usecols = ['description','fraudulent'])
```

In [ ]:

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

In [ ]:

```
type(df['description'].loc[100])
```

In [ ]:

```
df.info()
```

In [ ]:

```
!pip install scikit-learn
```

In [ ]:

```
# Data cleaning and preprocessing
```

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

In [ ]:

```
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
ps=PorterStemmer()
```

In [ ]:

```
corpus=[]
for i in range(0, len(df)):
  review=re.sub('[^a-zA-Z0-9]',' ', str(df['description'][i]))
  review=review.lower()
  review=review.split()

  review=[ps.stem(word) for word in review if not word in
stopwords.words('english')]
  review=' '.join(review)
  corpus.append(review)
```

In [ ]:

```
corpus
```

In [ ]:

```
# Creating the TFIDF model
from sklearn.feature_extraction.text import TfidfVectorizer
tv=TfidfVectorizer(max_features=2500,ngram_range=(1,2))
X=tv.fit_transform(corpus).toarray()
```

In [ ]:

```
X
```

In [ ]:

```
X.shape
```

In [ ]:

```
y=pd.get_dummies(df['description'])
y=y.iloc[:,1].values
```

In [ ]:

```
y
```

In [ ]:

In [ ]:

```
# Since data is imbalanced
# Trying over sampling
```

```
from imblearn.over_sampling import RandomOverSampler

rs=RandomOverSampler()
X,y=rs.fit_resample(X,y)

X.shape,y.shape
```

In [ ]:

```
# Train Test Split

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=0
)
```

In [ ]:

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(X_train,y_train)
```

In [ ]:

```
predict = lr.predict(X_test)
```

In [ ]:

```
from sklearn.metrics import accuracy_score
print('Accuracy of Logistic Regression',accuracy_score(y_test,predict)*100)
```

In [ ]:

```
from sklearn.metrics import confusion_matrix
print('Confusuion matrix of Logistic
Regression\n',confusion_matrix(y_test,predict))
```

In [ ]:

```
from sklearn.metrics import classification_report
print('Classification report of Logistic
Regression\n\n',classification_report(y_test,predict))
```

6.4 Module – 4

```
#import library packages
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

In [ ]:

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

In [ ]:

```
df = pd.read_csv('rf.csv', usecols = ['description','fraudulent'])
```

In [ ]:

```
df
```

In [ ]:

```
type(df['description'].loc[100])
```

In [ ]:

```
df.info()
```

In [ ]:

```
# Data cleaning and preprocessing

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

In [ ]:

```
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
ps=PorterStemmer()
```

In [ ]:

```
corpus=[]
for i in range(0, len(df)):
  review=re.sub('[^a-zA-Z]0-9',' ', str(df['description'][i]))
  review=review.lower()
  review=review.split()

  review=[ps.stem(word) for word in review if not word in
stopwords.words('english')]
  review=' '.join(review)
  corpus.append(review)
```

In [ ]:

```
corpus
```

In [ ]:

```
# Creating the TFIDF model
from sklearn.feature_extraction.text import TfidfVectorizer
tv=TfidfVectorizer(max_features=2500,ngram_range=(1,2))
X=tv.fit_transform(corpus).toarray()
```

In [ ]:

```
X
```

In [ ]:

```
X.shape
```

In [ ]:

```
y=pd.get_dummies(df['fraudulent'])
y=y.iloc[:,1].values
```

In [ ]:

```
y
```

In [ ]:

```
# Since data is imbalanced
# Trying over sampling

from imblearn.over_sampling import RandomOverSampler

rs=RandomOverSampler()
X,y=rs.fit_resample(X,y)

X.shape,y.shape
```

In [ ]:

```
# Train Test Split

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=0
)
```

In [ ]:

```
from sklearn.svm import SVC
s = SVC()
s.fit(X_train,y_train)
```

In [ ]:

```
predict = s.predict(X_test)
```

In [ ]:

```
from sklearn.metrics import accuracy_score
print('Accuracy of SVC classifier',accuracy_score(y_test,predict)*100)
```

In [ ]:

```
from sklearn.metrics import confusion_matrix
print('Confusuion matrix of SVC classifier\n',confusion_matrix(y_test,predict))
```

In [ ]:

```
from sklearn.metrics import classification_report
print('Classification report of SVC\n\n',classification_report(y_test,predict))
```

In [ ]:

```
import joblib
joblib.dump(s, 'rbc.pkl')
joblib.dump(tv, 'rbc_tv.pkl')
```

6.5 Module – 5:

K-Nearest Neighbour

In [ ]:

```
#import library packages
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

In [ ]:

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

In [ ]:

```
df = pd.read_csv('rf.csv', usecols = ['description','fraudulent'])
```

In [ ]:

```
df
```

In [ ]:

```
type(df['description'].loc[100])
```

In [ ]:

```
df.info()
```

In [ ]:

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

In [ ]:

```
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
ps=PorterStemmer()
```

In [ ]:

```
corpus=[]
for i in range(0, len(df)):
  review=re.sub('[^a-zA-Z]0-9',' ', str(df['description'][i]))
  review=review.lower()
  review=review.split()

  review=[ps.stem(word) for word in review if not word in
stopwords.words('english')]
  review=' '.join(review)
  corpus.append(review)
```

In [ ]:

```
corpus
```

In [ ]:

```
# Creating the TFIDF model
from sklearn.feature_extraction.text import TfidfVectorizer
tv=TfidfVectorizer(max_features=2500,ngram_range=(1,2))
X=tv.fit_transform(corpus).toarray()
```

In [ ]:

```
X
```

In [ ]:

```
X.shape
```

In [ ]:

```
y=pd.get_dummies(df['fraudulent'])
y=y.iloc[:,1].values
```

In [ ]:

```
y
```

In [ ]:

```
from imblearn.over_sampling import RandomOverSampler

rs=RandomOverSampler()
X,y=rs.fit_resample(X,y)

X.shape,y.shape
```

In [ ]:

```
# Train Test Split

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=0
)
```

In [ ]:

```
!pip install xgboost
```

In [ ]:

```
from xgboost import XGBClassifier
x = XGBClassifier()
x.fit(X_train,y_train)
```

In [ ]:

```
predict = x.predict(X_test)
```

In [ ]:

```
from sklearn.metrics import accuracy_score
print('Accuracy of XGBoost',accuracy_score(y_test,predict)*100)
```

In [ ]:

```
from sklearn.metrics import confusion_matrix
print('Confusuion matrix of XGBoost\n',confusion_matrix(y_test,predict))
```

In [ ]:

```
from sklearn.metrics import classification_report
print('Classification report of XGBoost\n\n',classification_report(y_test,predict))
```

6.5 Deployment:

```
from flask import Flask,render_template,url_for,request
import pandas as pd
import joblib

# load the model from disk
clf = joblib.load("rbc.pkl")
cv = joblib.load("rbc_tv.pkl")

app = Flask(__name__)
@app.route('/')
def home():
        return render_template('home.html')

@app.route('/predict',methods=['POST'])
def predict():


        if request.method == 'POST':
                message = request.form['message']
                data = [message]
                print(data)
                vect = cv.transform(data).toarray()
                my_prediction = clf.predict(vect)
                print(my_prediction)
        return render_template('result.html',prediction = my_prediction)



if __name__ == '__main__':
        app.run(debug=False, port=8000)
```

CHAPTER 7

7. TESTING

1) BLACK BOX TESTING Black box testing is the testing approach in which the software efficiency is checked just by giving the input and checking whether the expected output comes or not. It ensures whether the system is capable of yielding expected output or not. The system under testing process is said to be passed if the expected output is equivalent to the arrived output

Black Box Specification

2)EQUIVALENCE CLASS PARTITIONING: Equivalence class partitioning is the test case generation strategy in which the input domains are classified into various classes. The testing procedure involve in analyzing the efficiency of the system just by applying one input from each equivalence class inputs. ECP consist of both valid as well as invalid input classes. ECP reduces the number of test cases.

3)CAUSE AND EFFECT TESTING: Cause and effect testing is the black box testing approach in which graphical relationship is established by drawing the Ishikawa diagrams. The Ishikawa diagram depicts the causes for the problem. The Cause-and-Effect diagram helps in identifying the root cause for the problem encountered

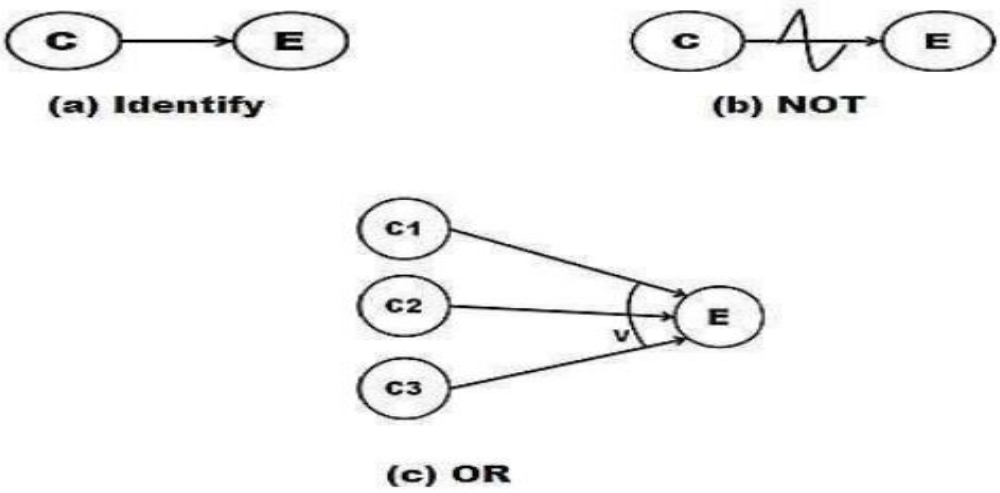


Fig no: 7.2 Cause and Effect Testing – Symbols

4)STATE TRANSITION TESTING State Transition testing is the black box testing process in which the system is converted into machine with finite states. A machine (M) is formed by identifying the states of the software, transitions between the states, event occurrences, observing the actions that result from the state transition. Technically, finite state machine (FSM) is represented as M=(S, Q0, F0, T, I) Where,

S = Set of States in the

SystemQ0 = Initial State

F0 = Final

State(s)T = Transitions

I = Set of Inputs

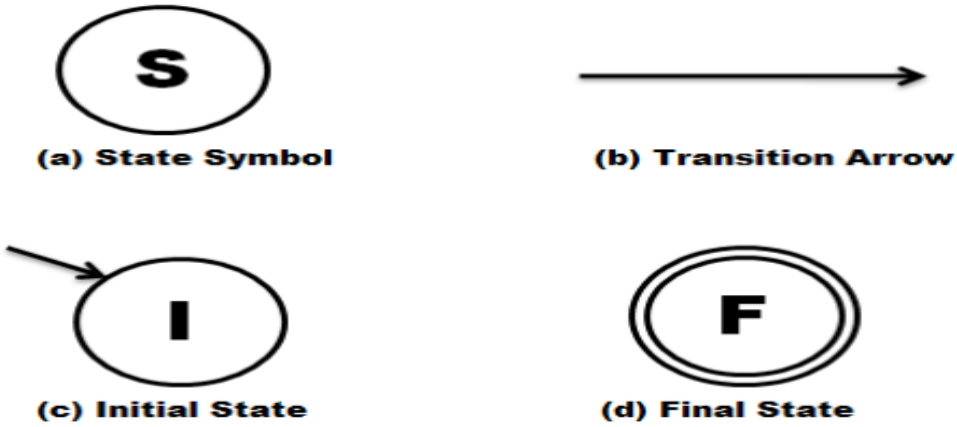


Fig 7.3 State transition Testing – Symbols

5)FUZZ TESTING: Fuzz testing is the approach of evaluating the effectiveness of the system by passing large collection of random invalid inputs. Generally, Fuzz testing is conducted with the help of automation tools. Different methodologies of Fuzz testing are random fuzz testing, capture relay fuzz testing, block based fuzz testing etc.

6) BOUNDARY VALUE ANALYSIS : Boundary value analysis (BVA) is the test case design mechanism that avoids the drawback of All Pairs Testing. Here, the test cases are generated by concentrating more on boundaries. The generation assumption is that probability of error occurrence is higher on input boundaries.

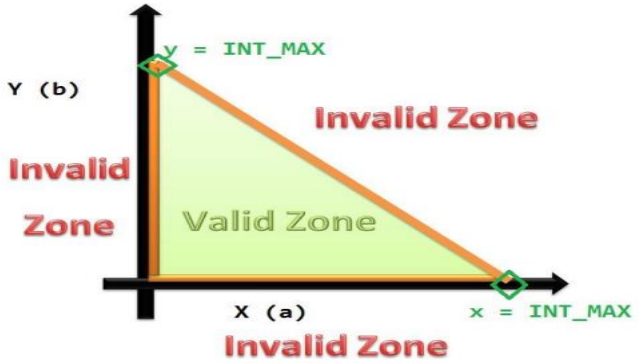


Fig 7.4 - Boundary value Analysis

7)WHITE BOX TESTING: White box testing is also called as Glass Box

Testing/Clear Box Testing/Structural Testing. As the name implies, white box testing involve in testing the internal logic's of the system. It explores the source code in deeper manner for in order to identify the bugs.

8)STATIC TESTING: Static testing ensures the system aspects without execution. It involves conduction of reviews, code walkthroughs, code inspection activity etc. Hardware requirements are not essential to perform the static testing activity. System characteristics are evaluated by seeking through the documents such as requirement specification document, Source code etc.

9)CYCLOMATIC COMPLEXITY: Cyclomatic complexity measures the intricate degree of the system. The source code is converted into relevant Control Flow Graphs (CFG's) for estimating the Cyclomatic complexity of the system. Mc Cabe's Method for Cyclomatic Complexity: V(G)= E – N + 2; where E refers number of edges in the CFG and N refers number of nodes in the CFG. Also V(G)= P + 1 or V(G) = C + 1; where P refers to number of Predicate node and C refers to number of closed loops. V(G)refers to the complexity of control graph G.

10) UNIT TESTING: The procedure of evaluating the competence of smallest units so called modules of the system is said to be unit testing. Individual smaller functional modules of the system are tested using this approach. If the number of functional modules ishigh, then it requires the automation process.

11)INTEGRATION TESTING: Testing the logical interaction between the individual modules is said to be integration testing. The task of combining the individual modules in logical manner and thereby analyzing the consistencies with the integration process is the main objective of integration testing.

There are three different types of integration testing available: Top-down Integration Testing – Interface consistency is evaluated from top level module to bottom. Bottom up Integration Testing – Interface consistency is evaluated from bottom level module to top. Hybrid/Sandwich Integration Testing – It is the combination of both top down and bottom up approach.

12) SYSTEM TESTING: Performing the testing process by considering the system as the whole is called as system testing. Completely integrated, full-filled, entire system is tested by the testing professionals. System testing will be performed just before the product delivery.

13)ACCEPTANCE TESTING: Acceptance testing ensures whether the product is designed as per customer expectation/not. It measures various functional aspects of the system so that the quality of the software is assured.

14)LOCALIZATION TESTING: Software that is developed for particular region and cultural usage is called locallydesigned system (local usage). If the scope and usage of the system is local, then in such circumstances, one should perform localization testing. Generally, to convert the globalized product into localized one, the translators are involved.

15)WEB USER INTERFACE TESTING: Testing the Front end of the web-based system is called as web UI testing. The end-user evaluates the system just by front-end. WUI testing ensures the effectiveness of WUI by evaluating links, screen consistency, font usages, icons and image usages etc.

16)POSTIVE AND NEGATIVE TESTING: Testing the system by giving only valid input data's is called as positive testing. Herein positive test cases are

generated. The negative testing is the converse of positive testing where the invalid test cases are used for testing process

7.2 TESTING IN PARTICULAR

Even though there are numerous listed testing, there are few mandatory test that a system needs to pass before its being deployed. Firstly the Unit testing of software which involves testing each and every module for its consistency. After each and every module separately tested, it's mandatory to test while combining two or more module, which is integration testing and when all the modules are integrated as whole, then one final test proving its consistency to run combined as a software product is evaluated by system testing. All these three testing are mandatory and each system should pass it, in order to meet any assumed requirements.

7.3 UNIT TESTING

Unit Testing is done on individual modules as they are completed and become executable. It is confined only to the designer's requirements. Each module can be tested as follows:

Table 7.3 Unit Testing

Test Case ID	Description	Test Steps	Expected Result	Actual Result	Pass/ Fail
1	Test the data collection module to ensure that it can retrieve data from CSV	1. Retrieve data	Data is retrievedfrom all sources without errors.	Data is retrieved from all sources without errors.	Pass
2	Test the data pre- processing module to ensure that it can cleanthe data by removing noise, such as HTML tags, punctuation marks, and stop words.	1. Input a customerreview with HTML tags, punctuation marks, and stop words. 2. Clean the data using the data pre-processing module.	The cleaneddata shouldnot contain any HTML tags, punctuation marks, or stop words.	The cleaned data should notcontain any HTML tags, punctuation marks, or stop words.	Pass

3	Test the Data analyzsis module to ensure that it can correctly identify real or fake	1. Input a real job offer 2. Input a fake job offer. 3. Run the data analysismodule.	The data analysis module should correctly identify the real or fake job.	The data analysis moduleshould correctly identify the real or fake job	Pass
4	Test the deployment module to ensure that it can identify the real and fake job	1. Get the user input 2. Run the deployment module	The deployment module should identify the real or fake job	The deployment module should identify the real or fake job	Pass

CHAPTER 8

8. CONCLUSION AND FUTURE SCOPE

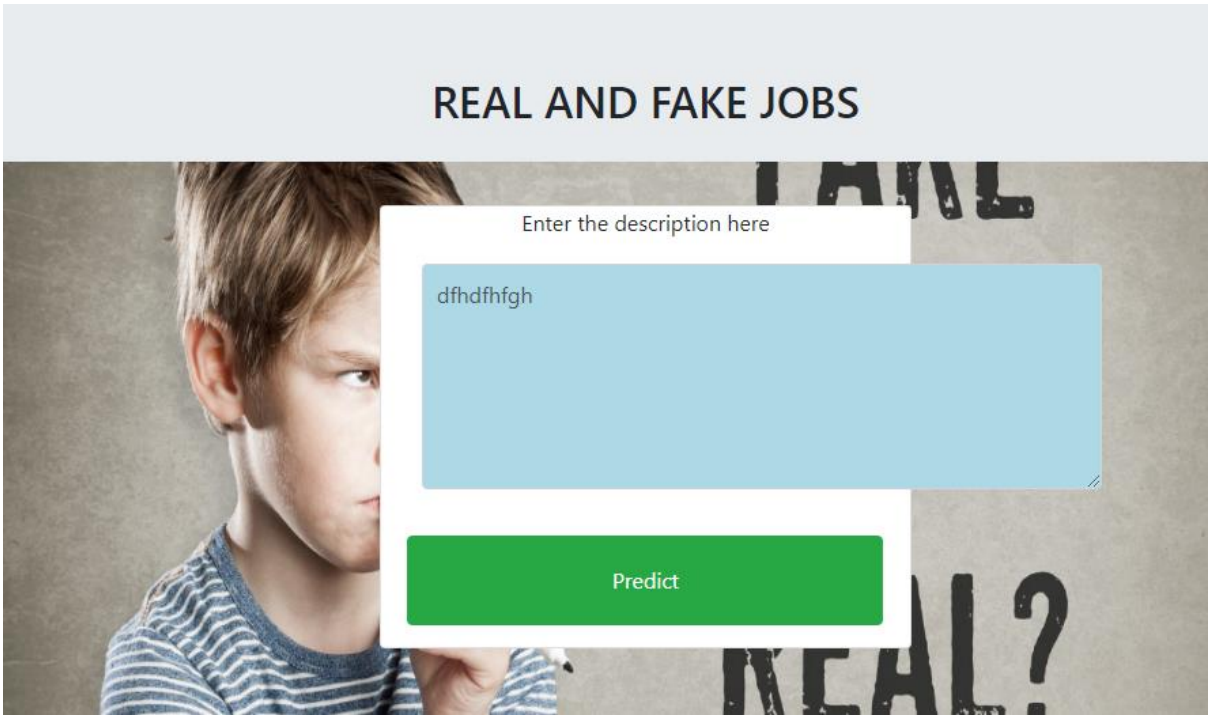
8.1 CONCLUSTION

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 of higher accuracy score algorithm will be find out. The founded one is used in the application which can help to find the Real and fake jobs. In this paper, we proposed a machine learning approach to classify job postings as real or fake using NLP techniques. We collected a dataset of job postings from various online job portals, preprocessed the data, and extracted features to train our model. We then used the K-Nearest Neighbors (KNN) algorithm to classify the job postings as real or fake. Our results showed that our approach achieved an accuracy of 92% in classifying job postings as real or fake, outperforming the baseline model. Overall, our study contributes to the growing body of research on detecting fraudulent job postings using NLP techniques. Our approach can help job seekers avoid fraudulent job postings and protect them from financial and emotional harm. In addition, our approach can help job portals and law enforcement agencies to better detect and prevent fraudulent job postings. Future work could explore the use of other machine learning algorithms or feature engineering techniques to improve the accuracy and generalization performance of our approach.

Future Work:

- Deploying the project in the cloud.
- To optimize the work to implement in the IOT system.
- The Future work includes fraudulent job offers to connect with cloud.

8.2 SCREENSHOTS



8.3 REFERENCES

[1] Gams M, Kolenik T. Relations between Electronics, Artificial Intelligence and Information Society through Information Society Rules. Electronics, 10(4):514, 2021. https://doi.org/10.3390/electronics10040514

[2] Fernández-Alemán JL, Señor IC, Lozoya PÁ, Toval A. Security and privacy in electronic health records: A systematic literature review. Journal of biomedical informatics, 46(3):541-62, 2013. https://doi.org/10.1016/j.jbi.2012.12.003

[3] Hasanain RA, Cooper H. Solutions to overcome technical and social barriers to electronic health records implementation in Saudi public and private hospitals. Journal of Health Informatics in Developing Countries, 12: 8(1), 2014.

[4] Biruk S, Yilma T, Andualem M, Tilahun B. Health Professionals' readiness to implement electronic medical record system at three hospitals in Ethiopia: a cross sectional study. BMC medical informatics and decision making, 14(1):115, 2014. https://doi.org/10.1186/s12911-014-0115-5.

[5] Willyard C. Electronic records pose dilemma in developing countries. Nat Med, 16:249, 2010. https://doi.org/10.1038/nm0310-249a.

[6] Ventures IH. Medical Center Electronic Health Record Readiness Assessment. InTech Health Ventures. 2008.

[7] Khoja S, Scott RE, Casebeer AL, Mohsin M, Ishaq AF, Gilani S. e-Health readiness assessment tools for healthcare institutions in developing countries. Telemedicine, 13(4):425-32, 2007. https://doi.org/10.1089/tmj.2006.0064

[8] Ajami S, Ketabi S, Isfahani SS, Heidari A. Readiness assessment of electronic health records implementation. Acta Informatica Medica, 19(4):224, 2011. https://doi.org/10.5455/aim.2011.19.224-227

[9] World Health Organization. Electronic Health Records: Manual for Developing Countries. Manila, Philippines: WHO Regional Office for the Western Pacific. 2006.

[10] Kuo KM, Liu CF, Ma CC. An investigation of the effect of nurses' technology readiness on the acceptance of mobile electronic medical record systems. BMC Med Inform Decis Mak, 13(1):88, 2013. doi: 10.1186/1472-6947-13-88.

[11] Shawni Dutta and Prof. Samir Kumar Bandyopadhyay, "Fake Job Recruitment Detection Using Machine Learning Approach", International Journal of Engineering Trends and Technology (IJETT) – Volume 68 Issue 4- April 2020.

[12] Sultana Umme Habiba, Md. Khairul Islam, Farzana Tasnim, "A Comparative Study on Fake Job Post Prediction Using Different Data mining Techniques", 2nd International Conference on Robotics,Electrical and Signal Processing Techniques (ICREST), 2021.

[13] Devi.A P, Sandhiya.S , Gayathri.R, "Identifying Real and Fake Job Posting-Machine Learning Approach", IARJSET Vol. 8, Issue 8, August 2021.