DETECTING FAKE JOB POSTING USING ML CLASSIFICATIONS AND ENSEMBLE MODEL

A PROJECT REPORT

Submitted by

RACHANNA DEVA MURALI [Reg No: RA1811003040099]
HARSITA R [Reg No: RA1811003040122]
AADHARSH K PRAVEEN [Reg No: RA1811003040225]

Under the guidance of

Mrs. S. NIVEDITHA
(Assistant Professor, Sr. G,
Department of Computer Science and Engineering)

in partial fulfillment for the award of the degreeof

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

Department of Computer Science & Engineering
Vadapalani Campus,Chennai

MAY 2022

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that 18CSP109L project report titled "**Detecting Fake Job Posting Using Ml Classifications and Ensemble Model**" is the bonafide work of "**RACHANNA DEVA MURALI [Reg No: RA1811003040099], HARSITA R [Reg No: RA1811003040122]** and **AADHARSH K PRAVEEN [Reg No: RA1811003040225]**", who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

GUIDE
Mrs. S. Niveditha
Assistant Professor,
Senior Grade
Dept.of Computer Science & Engg
SRMIST,Vadapalani Campus

HEAD OF THE DEPARTMENT Dr. S Prasanna Devi, B.E.,M.E., Ph.D., PGDHRM.,PDF(IISc)
Professor
Dept.of Computer Science & Engg
SRMIST,Vadapalani Campus

INTERNAL EXAMINER

EXTERNAL EXAMINER

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Own Work Declaration Form

This sheet must be filled in and signed with date along with the student registration number, work will not be marked unless this is done

To be completed by the student:

Degree/ Course	**:**	B. Tech / CSE
Student Name	**:**	**Rachanna Deva Murali, Harsita R and Aadharsh K Praveen**
Registration Number	**:**	**RA1811003040099, RA1811003040122, RA1811003040225**
Title of Work	**:**	**Detecting Fake Job Posting Using ML Classifications and Ensemble Model**

We hereby certify that this work compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism**, as listed in the University Website, Regulations, and the Education Committee guidelines. We confirm that all the work contained in this project is my / our own except where indicated, and that we have met the following conditions:

- Clarity references / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc.)
- Given the sources of all pictures, data, etc. that are not my own.
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g., fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

We understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:
We are aware of and understand the University's policy on Academic misconduct and plagiarism and we certify that this assessment is our own work, except where indicated by referring, and that we have followed the good academic practices noted above.
If you are working in a group, please write your registration numbers and sign with the date for every student in your group

Signature of the Student(s)

ACKNOWLEDGEMENTS

We express our humble gratitude to **Dr. C Muthamizhchelvan**, Vice Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support. We extend our sincere thanks to **Our Dean Dr. C. V Jayakumar**, SRM Institute of Science and Technology, Vadapalani campus for his invaluable support. We wish to thank **Dr. S Prasanna Devi**, Professor & Head, Department of CSE, SRM Institute of Science and Technology, Vadapalani Campus for her valuable suggestions and encouragement throughout the period of the project work.

We are extremely grateful to our Class Coordinator, **Dr. S Manohar, Assistant Professor,** Department of CSE, SRM Institute of Science and Technology, Vadapalani Campus, for leading and helping us to complete our course.

Our inexpressible respect and thanks to my guide, **Mrs. S. Niveditha**, **Assistant Professor, Sr. G** Department o f **C S E ,** SRM Institute of Science and Technology, Vadapalani Campus for providing us an opportunity to pursue our project under her mentorship. She provided us the freedom and support to explore the research topics of our interest.

We sincerely thank our Management, staff, and students of the Department of **C S E ,** SRM Institute of Science and Technology, Vadapalani Campus who have directly or indirectly helped our project. Finally, we would like to thank our parents, our family members and ourfriends for their unconditional love, constant support and encouragement.

Aadharsh K Praveen

Harsita R

Rachanna Deva Murali

ABSTRACT

In our project, we create a fraudulent checker tool to detect fake job postings using NLP (Natural Language Processing) and ML approaches (Random Forest Classifiers, Logistic Regression, Support Vector Machines, and XGBoost Classifiers). These approaches will be compared and then combined into an ensemble model which is used for our job detector. The goal is to predict actual or fake job prediction outcomes with the maximum accuracy using machine learning-based techniques. The dataset will be analysed using the Supervised Machine Learning Technique (SMLT) to capture numerous details such as variable identification, missing value treatments, and data validation analyses. The complete dataset will be cleaned/prepared, and data visualisation will be performed. The ensemble model is created at the end using ML Algorithms like XGBoost, SVM, Logistic Regression, and Random Forest Classifier by choosing 4 of the best contributing features. The model produced at the end will be implemented in a Flask application for demonstration.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

RFC	-	Random Forest Classifier
LR	-	Logistic Regression
DT	-	Decision Tree
MSE	-	Mean Square Error
EDA	-	Exploratory Data Analysis
CD	-	Company Description
SVM	-	Support Vector Machine
JD	-	Job Description
OS-ELM	-	Online Extreme Learning Machine
ELM	-	Extreme Learning Machine
FCL	-	Full Center Loss
ACL	-	Angle Center Loss
BOW	-	Bag Of Words
MNB	-	Multinomial Naïve Bayes
KNN	-	K- Nearest Neighbors
OOB	-	Out Of Bad Dataset
TVM	-	Term Vector Model
ANN	-	Artificial Neural Network
MLP	-	Multi-Layer Perceptron
NLP	-	Natura Language Processing
AI & ML	-	Artificial Intelligence & Machine Learning
TF-IDF	-	Term Frequency — Inverse Document Frequency

CHAPTER 1

INTRODUCTION

Many companies, whether well-known or new, prefer to post job applications online. Since it is fast and can easily reach a wide variety of capable applicants. But because of this, lots of scams also arise. Identifying a fake job application depends on many factors. Sometimes scams can be easily identified if the employer asks for money from the employee. But there are also tricky scams that will make the user give their details which are more dangerous than losing money. Since anyone can view these job postings, freshers who are gullible might fall for these scams easily.

So, to prevent this, the Machine Learning approach is used to classify, whether the job application is real or fake. Using our ensemble model, we can take a lot of factors into consideration. The primary factor is the description of the job. The other factors include company profile, benefits, requirements, etc. With these factors, we can decipher whether the job is real or fake.

1.1 PROBLEM STATEMENT

Work-from-home jobs have long been a target for scammers, with a 300 percent surge in hiring scams prior to 2017, and another increase in frauds until 2020. They have, however, become even more vulnerable targets in the aftermath of the COVID-19 issue. Because of the coronavirus outbreak, many people have lost their jobs. Finding a new work can be challenging, especially because many non-essential firms throughout the world have had to cut hours or reduce staff, resulting in mass layoffs. Scammers are acutely aware of the fact that some job searchers are in severe need of cash.

In recruitment, there is the good, the bad, and the ugly. From fraudsters to copy-paste connoisseurs, some of them are even accountable for tarnishing the reputation of "legitimate" recruiters. Scams involving job searches and recruiting are not uncommon on LinkedIn. If you receive a communication from someone expressing interest in hiring you for a position at their organization, it's most likely a real recruiter contacting you about a legitimate job. However, there is always the chance that the

recruitment is a hoax. Scammers use LinkedIn to reach out to targets, knowing you're more likely to fall for the scam just because the message is coming through a reputed platform like LinkedIn. However, to be on the safer side and avoid falling prey to scams, it is advised that we treat every unsolicited offer as a job scam regardless of where it comes from and how well known the platform happens to be.

Some recruitment agencies may also use strategies that defy work ethic, like advertising phantom roles to make it seem more appropriate on the outside when they already have a new employee in mind. Some companies collect CVs to analyze the market while some use that as a disguise to operate dubious businesses like illegally selling the data they collect. Pyramid marketing is illegal and has no basis in real commerce. For someone to make money with a pyramid marketing scheme, someone else must lose funds.

Verifiable Information definitely plays an important role in determining the authenticity of the job posting. You may have believed you landed your perfect job, but after closer inspection, you can't locate any information about the organization. If you cannot confirm contact information, location, website, or employees, then you may have fallen prey to fraudulent recruitment. In this day and age, genuine businesses will have an internet presence as well as some social media engagement and the lack thereof could also signify that they may not be authentic.

Therefore, to prevent such scams, the Machine Learning approach is used to classify whether the selected job application is real or fake. Using our ensemble model, we can consider several factors. The primary factor is the description of the job. The other factors include company profile, benefits, requirements, etc. With these factors, we can decipher whether the job posting is real or fake.

1.2 DATA SCIENCE AND SCIENTISTS

Data science is a discipline that employs scientific approaches, algorithms, processes, and systems to extract information from many types of unstructured and structured data, as well as to apply that knowledge to a myriad of different areas. As an alternative to computer science, Peter Naur suggested data science. The first conference to promote data science was organised by the International Federation of Classification. The definition, however, was in flux. D.J. Patil and Jeff Hammerbacher

invented the term "data science" in 2008. It has become one of the most popular occupations in the sector in less than a decade. Data science is the study of deriving usable insights from data by integrating topic knowledge, computer skills, and an understanding of arithmetic and statistics [1]. Data science is a mix of arithmetic, business skills, tools, algorithms, and a variety of machine learning approaches that aid in the discovery of hidden insights or patterns in raw data that may be utilised to make critical business decisions.Data scientists explore and analyze which questions must be answered and where relevant data might be found. They are well-versed in business and analytical abilities, as well as the ability to mine, clean, and display data. Data scientists are employed by businesses to collect, organize, and evaluate enormous volumes of unstructured data.

1.3 ARTIFICIAL INTELLIGENCE

Any computer or program that mimics human intelligence is known as artificial intelligence (AI). By learning in the same way that humans do, AI tries to emulate the human brain. AI improves and learns problem-solving skills. Machine intelligence, as opposed to natural intelligence demonstrated by living beings, is referred to as AI. The topic is described as the study of "intelligent agents," which are any systems that sense their environment and take actions to enhance their chances of attaining their goals, according to leading AI textbooks. AI experts, on the other hand, contested this definition, which uses the word "artificial intelligence" to refer to robots that mimic "cognitive" functions performed by humans, such as "learning" and "problem-solving."

AI is used in intelligent search engines like Google, recommendation systems like Netflix, self-driving vehicles like Tesla, speech recognition like Siri and Alexa, and strategic video game systems like chess at the highest levels. When robots become more capable, jobs formerly thought to require "intellectual aptitude" are removed from the AI equation. Take, for example, optical character recognition. Artificial intelligence has gone through a series of ups and downs since its inception as an academic research project in 1956 [2], with new approaches, success, and renewed funding.

Throughout its history, AI research has investigated and rejected a wide range of approaches, including brain mimicry, modelling human problem-solving abilities, formal logic, enormous information libraries, and animal behavior mimicry. In the early decades of the twenty-first century, highly quantitative statistical machine learning dominated the field, and this approach has shown to be extremely effective, assisting in the resolution of many difficult challenges in industry and academia.

The numerous subfields of AI research are centred on specific goals and methodologies. Logic-based information processing, data scheduling, model training, natural language processing (NLP), sensing, and the capacity to move and manipulate objects are all traditional AI research aims. One of the program's long-term goals is general intelligence (the capacity to solve any issue). AI researchers utilise search and optimization algorithms, logic programming, convolutional neural networks, and statistics, probability, and economics-based approaches to address these issues. Computer science, linguistics, psychology, and a variety of other subjects are all used in AI.

As the hype around AI has grown, suppliers have been scrambling to show how AI is integrated into their goods and services. When we talk about AI, we're talking about a single component called machine learning. For constructing and training machine learning algorithms, AI requires a foundation of specialized hardware and software. Although no single scripting language is connected with artificial intelligence, Python, R, and Java stand out. AI systems frequently consume large volumes of labelled training data, evaluate, and recognize patterns in the data, and then use these structures to predict future states.

A chatbot given samples of text dialogues may learn to generate lifelike discussions with people by studying millions of cases, while an image recognition computer may learn to recognize and describe things in images by evaluating millions of instances. AI typically includes three processes known as learning, which refers to the human brain, reasoning, and self-correction.

1.4 LEARNING PROCESS

Learning processes are an aspect of AI programming that is involved with receiving data and developing rules for converting it into valuable knowledge for the process. Algorithms are rules that provide computer equipment with stage-by-stage instructions for executing a certain activity.

1.5 REASONING PROCESS

Reasoning processes are an aspect of AI programming that is involved with determining the optimum

method to achieve a given goal.

1.6 SELF-CORRECTION PROCESS

This element of AI programming aims to fine-tune algorithms on a frequent basis in order to ensure that they give the highest level of accuracy.

1.7 NATURAL LANGUAGE PROCESSING(NLP)

Machines can read and comprehend human language thanks to natural language processing (NLP). Natural language is a user interface that is generated directly from human-written sources, such as news agency articles, and it is theoretically possible with an adequate natural language processing system. Web scraping, text mining, question answering, and machine translation are all examples of simple natural language processing applications. To generate syntactic representations of text, some recent methodologies use word co-occurrence frequencies. "Term spotting" search algorithms are well-known and adaptable, but they are also erroneous; for example, a search for "dog" may only return papers that contain the keyword "dog," but articles containing the keyword "poodle" may be overlooked.

Lexical affinity approaches assess the emotional content of a material by searching for keywords such as "accident." [3] In many circumstances, modern statistical NLP algorithms may incorporate all of these tactics, as well as others, and get correct answers at the page or paragraph level. Beyond semantic NLP, the ultimate goal of "narrative" NLP is to contain a thorough understanding the basics of intuitive thinking. Deep learning systems built on transformers may also be able to generate coherent text by 2019.

1.8 MACHINE LEARNING

Machine learning is a method of projecting the future based on historical data. Machine learning (ML) is an AI approach that allows machines to understand and learn without having to be explicitly programmed. Machine learning is concerned with the production of data-adaptive computer programs, as well as the principles of machine learning, such as the construction of a simple learning algorithm

in Python. In the training and prediction phase, specialized algorithms are used. It sends the training sets to an algorithm, which then uses the training data to create predictions using new test data.

Machine learning can be broken down into three categories. Learning can be classified into three categories: supervised, unsupervised, and reinforced. To learn data that must first be tagged by a person, a supervised learning algorithm is given both the inputs and the accompanying labelling. In unsupervised learning, there are no labels. The deep learning model was given access to it. This method must determine how the data in the input is clustered. Finally, reinforcement learning interacts dynamically with its environment and receives positive or negative feedback in order to improve its performance. Data scientists use a number of machine learning algorithms to identify relationships in Python that lead to valuable insights.

Based on how they "learn" about data to make predictions, these algorithms can be classified into two categories: supervised and unsupervised learning. Classification is a strategy for guessing the classification of supplied data sets. Classification predictive modelling is the process of predicting a map function between input variables (X) and different output variables (y).

$$y=f(X)$$

Where y is the desired output and X is the input variable. We map the input and output in a nonlinear fashion.

In statistics and machine learning, classification is an important method in which a computer software learns from data input and then uses that learning to categorize new observations. This data collection might be classified as bi-class (for example, determining whether the mail is spam or non-spam) Examples include voice recognition, fingerprint recognition, identity verification, document categorization, as well as other classification issues. [4] The great majority of practical machine learning applications make use of Supervised Machine Learning. The objective is to calculate the mapping function sufficiently to foresee the target value (y) for new input data (X). Supervised machine learning approaches include multiple regression models, multi-class classification, SVMs and Decision Trees. The data required for training the algorithms for supervised learning must be labelled with correct answers before testing the model.

Problems with classification are a subclass of supervised learning algorithms. The goal of this challenge is to develop a basic model that predicts the value of dependent variable attribute based

solely on the attribute variables. The main difference between the two tasks is that in categorical classification, the dependent feature is numerical. A classification model attempts to infer something from observable data. The classifier would attempt to anticipate the results of one or more outcomes depending on the values of one or more inputs. The problem is referred to as a classification problem when the outcome is a categorization, such as "red" or "blue."

1.9 EXISTING SYSTEMS

Before choosing out model, we have researched about existing systems. This uses the different methods, but the end goals are similar. Upon further research we have found out why these methods are lacking.

1.9.1 NAÏVE BAYES CLASSIFIER

Naïve Bayes is a technique for classifier construction. For training such classifiers, there is no specific algorithm, but rather a combination of algorithms. Naïve Bayes Classifier assumes all values of one feature are independent of values of other features. It is often used to build machine learning models that can make quick predictions. [5] A probabilistic classifier predicts based on the likelihood of an item. Assumes that all predictors are independent, which is seldom the case in real life. It is confronted with the zero-frequency issue.

For example, each feature is considered to contribute independently to the probability that the vegetable is a carrot, regardless of any possible correlations between the color, shape, and height features. In practical applications of Naïve Bayes Classifier, they use Maximum Likelihood, it’s the probability of maximum chances of the entity being a class. An advantage of this is that it requires small training data for classification. By looping a supervised learning algorithm, you can create a semi-supervised training algorithm that can learn from both labeled and unlabeled data.

Examples of Naïve Bayes Classifier:
Person Classification - If a person is male or female depending on features like height, weight, and foot size.
Document classification - Classifying documents by their content. Like e-mail spam detection.

1.9.2 ADABOOST CLASSIFIER

AdaBoost is a boosting technique. It is an ensemble model with 2 steps. First, a classifier is fit to the dataset. It then fits the same classifier again to the dataset but substantially alters the weight of incorrectly classified instances such that subsequent classifiers focus on complicated situations. To transform a bad classifier that is just marginally better than a random guess into an excellent classifier. Adaptive Boosting is so named because the weights are redistributed to each instance, with bigger weights applied to incorrectly assigned instances. [6] A high-quality dataset is required. Before implementing an Adaboost algorithm, it is necessary to prevent noisy data and outliers. It is easier to use with less need for tweaking parameters but AdaBoost is not prone to overfitting. Adaboost technique learns progressively, it is important to note that you have quality data. It is extremely sensitive to Noisy data so caution must be taken when using AdaBoost.

1.9.3 K-NEAREST NEIGHBHOURS CLASSIFIER

A Supervised Machine Learning approach is used in the K-Nearest Neighbor algorithm. KNN maintains all data and compares it to newer data to determine the degree of similarity. It assumes that the input data and cases in the dataset are comparable and, as a result, places the new instance inside a category that seems to be closest to the existing categories. [7] KNN needs a massive amount of memory to store all the data. A supervised machine learning algorithm that can deal with both classification and regression problems. To resample datasets and fill in missing values. It needs a large amount of memory in order to store all of the training data.

Example:

Predicting animals - If features of animals are labelled. It compares the input data with the data set, then it gives the output based upon the calculation of the K value and nearest neighbors.

1.9.4 MULTI-LAYER PERCEPTRON CLASSIFIER

MLP is a class of ANN (Artificial neural network). MLP has at least 3 basic layers, Input, Output, and hidden layer. It uses a supervised learning technique named backpropagation for training the data. It has multiple layers and non-linear activation, so it is different from linear perceptrons. MLP contains many perceptrons that are arranged like layers. [8] Perceptrons are like a special scenario of artificial neurons that use a certain threshold activation function. MLP was popular in finding applications like speech and image recognition, machine translation, etc.

1.9.5 DECISION TREE CLASSIFIER

The decision tree has nodes that specify an attribute, and each branch denotes the one in many values for that attribute. Leaf represents the class labels. The algorithm is that the classification starts at the root node and each node splits into two or more subtrees according to a condition, at the end a new node is created. [9] This process carries on till all data is classified. It works in a top-down manner. There are lots of possibilities to measure the split of the subtrees. When the user is missing one or two inputs, the model is capable of identifying whether the job posting is fake or real.

CHAPTER 2

LITERATURE REVIEW

A review of the literature is a piece of writing that attempts to summarizes the most essential components of current understanding of methodological procedures as they relate to a specific topic. It is a secondary source that discusses published information in a certain subject area, as well as expertise in that subject field during a specified time period. Its ultimate goal is to keep the reader up to date on the reviewed literature, and it also serves as a foundation for other objectives, such as future studies that may be required in the field. It may just be a list of references that appears before a proposed investigation. It usually follows a pattern and includes summary as well as synthesis. We write a summary to help us understand a long text more quickly while yet keeping the meaning. It could provide a new perspective on existing material, combine new and old viewpoints, or track the field's intellectual growth, including major debates. Depending on the context, a review of the literature may examine the materials and direct the audience to the most current or relevant ones.

2.1 LITERATURE REVIEW 1

Title: Predicting of Job Failure in Compute Cloud Based on Online Extreme Learning Machine: A Comparative Study [37]

Author: C. Liu, J. Han, Y. Shang, C. Liu, B. Cheng, and J. Chen

Year: 2017

Performs online job failure prediction on Google datasets to improve online prediction models and resource utilization by comparing various prediction models. OS-ELM is used to predict job status by collecting real-time data according to the sequence of arrival. It reduces storage space and resources used in the cloud by intelligently identifying job failure. Due to its fast-learning speed and good generalization it takes very little time to update the model while providing higher prediction accuracy and better false-negative performance when compared to other existing models. The training and testing time for SVM and ELM models are much shorter than OS-ELM’s and with better offline prediction performance. Accuracy, precision, and false-negative rate with prediction accuracy of 93% and updates the model in 0.01s

They are analyzing the different states a task could end up at. For example, it could be successful and go to finished state, the job could be killed, or evicted or the job could fail and resubmitted. There are lots of jobs are assigned to a particular system, and the system schedules the jobs according to its priority. They analyzed the status of jobs, from their research they have found that about 60% of the jobs are successful, and 40% are either evicted, failed, or gets killed, thus leading to job termination. ELM (Extreme Learning Machine) is a learning algorithm which can solve single hidden layer neural network. How ELM works is that it inputs random weights and offset and gets the output in output layer.

OS-ELM is a modification of ELM which has faster learning speed. Its an online incremental algorithm, it can deal with the sequential arrival of data. Unlike ELM, OS-ELM does not reuse learned data for updating the model. They are gathering the required data from google clusters [38]. The logs of job submission are gathered in real-time in cloud. The features are extracted from the pre-processed data.
The data from cloud are cleaned if:
The job didn't get executed after being submitted for 20 minutes.
The job hasn't been completed at the end of trace record.
The job finished before it has been scheduled.
Data went missing.
Job started before traces were recorded.

With the static characteristics the jobs are extracted in data preprocess, and this used as a feature vector. With the extracted features, the data is split into two parts, test data and training data in the ratio 1:3. The training data is used to train the model to predict the status of the recent jobs. If the model predicts that the job will be successful, then the job continues. If the model predicts it to be in termination state, the job terminates and resubmitted, waiting to be scheduled. OS-ELM model adapts the strategy of fast learning in predicting and model updating to provide low cost with best performance.
The method that is used to predict the state has two phases: initialization phase and the sequential learning phase [39, 40].

SVM is usually used for small-scale data for classification problems [41, 42]. SVM maps the problem to high dimensional space. The advantage of SVM is that global optimization, strong generalization

ability and small sample [43]. OS-SVM is an advanced version of SVM, which does online rapid incremental learning [44, 45]. This works in offline phase. When training the model, they have discovered that ELM is faster when compared to SVM and time taken for ELM training and testing is 100 times shorter when compared to OS-ELM. From this they can draw a conclusion that offline models work better than online models since offline uses large samples at a time to build the model, where online models use small samples at first and perform the model update training as data samples reach. However, online models time performance is better when compared to offline models.

From the testing and training they have observed that OS-ELM model is suitable for online predictions. The reasons are:

i) SVM and ELM are offline models which requires more storage space since the model needs to train from existing samples.
ii) OSELM has an advantage of learning faster, updating the model takes about 0.01s, where OS-SVM takes about 22.04s for updating.
iii) OS-ELM model is more stable when compared to OS-SVM because while updating the model OS-SVM needs to retain the samples that violated the conditions for next iteration, wherein for OS-ELM the hidden-layer weights and parameters are randomly chosen, and feed forward neural network is minimized.

Thus, in conclusion, time performance and prediction of OS-ELM is superior to other models. It can reduce the storage space by identifying job failure and reduce the resource wastage in cloud.

2.2 LITERATURE REVIEW 2

Title: Deep Representation Learning with Full Center Loss for Credit Card Fraud Detection [10]

Author: Z. Li, G. Liu, and C. Jiang

Year: 2020

Focuses on obtaining deep feature representations of legal and fraud transactions from the aspect of the loss function of a deep neural network. Uses Full Center Loss and Angle Center Loss on two big datasets of credit card transactions: one public Kaggle dataset and the other a private dataset. FCL considers both distances and angles among features and can comprehensively supervise deep representation learning. Feature engineering is used to extract informative features of transaction

behaviors and ACL is an improved SL used here. Though this model ensures stability it does not address the concept of drift problems. The performance metrics used to evaluate the model here are accuracy, precision, and recall and achieved an accuracy of 82.2%

2.3 LITERATURE REVIEW 3

Title: Fake Job Prediction using Sequential Network [11]

Author: D. Ranparia, S. Kumari and A. Sahani

Year: 2020

Uses Natural Language Processing to analyze sentiments and patterns in job postings on LinkedIn using Beautiful Soup. Trained the model as a Sequential Neural Network using the GloVe algorithm and uses the EMS CAD dataset. The Global Vector Model Algorithm used is very realistic and can determine if the job is real or fake just by extracting the job description and feeding it into the model. Sentiment analysis can be improved by also implementing Word2Vec along with the GloVe algorithm. The model was compiled for 10 epochs with a batch size of 64 and achieved 97.58% validation accuracy.

2.4 LITERATURE REVIEW 4

Title: Automatic Detection of Online Recruitment Frauds: Characteristics, Methods, and a Public Dataset [12]

Author: Videos, Sokratis, Constantinos Kolias, Georgios Kambourakis, and Leman Akoglu

Year: 2017

Analyzes all possible aspects of employment scams by exploring EMS CAD dataset that contains real-life legitimate and fraudulent job recruitment ads. Trains 6 popular WEKA classifiers using a bag of words modelling and evaluates their performances to generate a ruleset which is then converted into binary feature vectors and tested against the same WEKA classifiers. The use of various analyses helps improve automated anti-scam solutions by ATS to train classifiers and gain a deeper knowledge of the problem's characteristics. The ruleset does not focus on user behavior, company & network data, and user-content-IP collision patterns and lacks graph modelling. Empirical Analysis, stop word filtering (excluding articles and prepositions) and used k-fold cross-validation strategy to evaluate the model.

The data set they are using is EMSCAD. For the data cleaning they take only the key words while removing the stop words [13] like “the, an, with, etc.…”. after this they have used another bow (bag of words) modelling to train WEKA classifiers [14]. For training and testing they take features named as job description, company profile, requirements, and benefits. After cleaning of data is done, they use machine learning algorithm named as ZeroR, OneR, Naives Bayes, J48 decision trees, random forest, and logistic regression. Then they compare the results.

With these rules they separate the dataset. This new model can scale up large dataset as it requires less storage. The model is again tested with the six classifiers and observed the results. The data was separated into training and test data using k-fold cross validation technique. In this new model almost, all classifiers could do better performance when compared to bow. All classifiers are increased in accuracy by 2%-13%. Only random forest showed decline of 0.5%. now they have chosen random forest classifier as their test. They have concluded mentioning that features that are related to company like short company profile or lack of company logo and profile are very effective. On the contrary, legitimate jobs are found with short descriptions.

As the final step they have tested the random forest classifier that has been trained on empirical ruleset against the unbalanced 17,880 data. The model showed an accuracy of 89.5%. the precision and recall score for non-fraudulent was 0.986 and 0.903 accordingly. But fraudulent has 0.282 and 0.751. Since the dataset is highly imbalanced, these results are to be expected. From observations they have concluded that with balanced dataset the model can produce an accuracy up to 90%. So according to the results, one fraudulent could be marked as non-fraudulent out of 10 fraudulent.

Their future goal is to employ graph modeling and explore connections between fraudulent job ads, companies, and users. They would also like to make their employment fraudulent tool to be of commercial use.

2.5 LITERATURE REVIEW 5

Title: Machine Learning and Job Posting Classification: A Comparative Study [15]

Author: Ibrahim M. Nasser1 and Amjad H. Alzaanin2

Year: 2020

In this paper, they have taken a text classification problem and compared various machine learning algorithms like SVM, Multinomial Naive Bayes, Decision Tree, K Nearest Neighbors, and Random Forest. The data set they sued contains real and fake jobs. They cleaned the dataset and pre-processed the text using TF-IDF for extracting the features. They split this data into two parts, train data and test data. Evaluation metrics used are precision, recall, f-measure, and accuracy. For each classifier, results were summarized and compared with others. The reason they are doing this research is to measure the performance of most used ML techniques on a text classification problem and prove a comparison between them.

The theory with Multinomial Naive Bayes is that to predict the probability of an event based on previous knowledge [16]. The Naive Bayes classifier proved their efficiency in text classification problems [17]. This works on solid independent assumptions; this means one assumption does not affect the others. Given n assumptions, this model makes 2n! independent assumptions. This also paves way on understanding each assumption separately since they are not dependent on each other [18]. There are two event models: the multi-variate Bernoulli, and the Multinomial Naïve Bayes (MNB). They are working on MNB. MNB gets the word frequency in documents [19].

Support Vector Machine (SVM) is a learning model by Vapnik. It can learn functions from labelled vectors [20]. The job of SVM is to find the optimal hyperplane by comparing the nearest two different classes data points, so it can generalize the training pattern [21].

Decision Tree Classifier (DT) is commonly used ML for classification and predictions. DT works with nodes and leaf. It has a tree structure. The top node is the root node, every other node is either a decision node or a leaf node. Decision node is the one which sets rules or tests are carried out on an attribute and classifies the value accordingly. While lead node just mentions the class of data that is it is classified into. Decision node can classify values into leaf node, or it can also have sub-tree. DT goes from top to bottom, starts at root node and works it way towards the leaf nodes [22].

K- Nearest Neighbors (KNN) is used mainly for its simplicity and efficiency [23-25]. It finds the Euclidean distance and assesses individual features. KNN predicts depending on a number that represents the nearest training example. It stores all the training samples and chooses the k nearest sample from the classified vectors and determines the class of the new data.

Random Forest (RF) is an ensemble of Decision Trees. So each DT provides one output and votes are conducted, the class with most votes becomes the prediction of RT [26]. To create a RF one has to create a bootstrapped dataset that is the same size as the original. To create it we randomly select data from original dataset. Few instances are missed from original dataset, these are called Out Of Bad Dataset (OOB), so each bootstrapped dataset has its own OOB. Then DT is created. The difference is that we choose only one feature and not the entire sample. This is the reason why RF is more effective than individual DT. In test phase, we take the OOB dataset and check if the predicted class is correct or not.

They have used Google Collaboratory to code. The data was uploaded and pre-processed. Then feature extraction is done. With this clean data we split it into train data (70%) and test data (30%). The dataset they are using is from Kaggle [27]. This has about 18K entries and 17 attributes and one class with binary values.

Now the features are getting extracted, this step is essential because its suitable for learning algorithms [28]. Converting text data to vector. Vector is a numeric value corresponding to each term appearing in a text [29]. They are using TF-IDF to convert text to vectors [30]. It takes the most frequent words in the document and use them as features vector. TF-IDF gives higher weight to important terms from the document. Now the data is converted to Term Vector Model (TVM).

The evaluation metrics they are using are accuracy, true positive rate, precision, recall, and F1 score. After they have trained the models, they constructed a confusion matrix and evaluation matrix. It is observed that Random Forest has highest accuracy of 98%. The second highest are SVM, DT, KNN with an accuracy of 97% and highest F1-score of 99.0. In terms of highest recall, its DT followed by RF and KNN. Meanwhile highest precision is achieved by MNB, SVM, and RF. Although MNB has lowest accuracy, it might be possible because of the state of nature stated in [31]. Finally, they have concluded that, for text classification problem Random Forest Classifier is best.

2.6 LITERATURE REVIEW 6

Title: Fake Job Recruitment Detection Using Machine Learning Approach [32]

Author: Samir Bandyopadhyay, Shawni Dutta

Year: 2020

They have used machine learning classification to avoid fake post for jobs in internet. The classifiers are tested and compared with each other, and they identify the best employment scam detection model. They can detect fake job posting in enormous number of job posts. They have used mainly two classifiers – single classifier and ensemble classifiers. With experiments its found that ensemble model is better at detecting scams when compared to single classifiers.

S

The classifier identifies fake job postings from a large set of advertisements and alerts the user. They consider a supervised machine learning to get this job done. The job of a classifier is to map the input data with target class while considering training data. The classifiers are broadly categorized to single classifier-based prediction and ensemble classifier-based prediction.

Single classifier-based prediction:

The single classifiers they have used are:

a. Naive Bayes Classifier
b. Multi-Layer Perceptron Classifier
c. K-nearest Neighbor Classifier
d. Decision Tree Classifier

Naive Bayes classifier is based on supervised learning which uses Bayes Theorem [33,34]. The classification made by this classifier is quite effective in practice even if its probability of estimation is inaccurate. Naïve bayes works well if features are independent of each other or features are completely functionally dependent. The accuracy of this classifier depends on the information loss due to the assumption of independent features. There is no single algorithm for training such classifiers, but rather a collection of algorithms. An advantage of this is that it requires small training data for classification. By looping a supervised learning algorithm, you can create a semi-supervised training algorithm that can learn from both labelled and unlabelled data.

Examples of Naïve Bayes Classifier:

- Person Classification:

  If a person is male or female depending on features like height, weight, and foot size.

- Document classification:

  Classifying documents by their content. Like e-mail spam detection.

By including optimized training parameters, multi-layer perceptron can be used as supervised learning tool. The number of hidden layers and number of nodes in each layer can differ based on the problem. The factor which decides the class is dependent on training data and network architecture [35]. MLP is a class of ANN (Artificial Neural Network). MLP has at least 3 basic layers, Input, Output, and hidden layerMLP was popular in finding applications like speech and image recognition, machine translation, etc.

K-Nearest Neighbor Classifiers is also known as lazy learner. The job of this classifier is to store all the training data. And when the test data is given the classifier maps the nearest class. The classifier takes k number of objects as nearest object. The challenge of this classification is that it relies on choosing the value of k [36]. KNN is also known as a lazy learner algorithm because it does not learn from the dataset, it just stores all values and finds similarities. The advantage is that it's simple to implement and it is robust to noise. It is directly proportional to the size of training data, it's more effective with large training data.

Decision Tree (DT) has a tree like structure using nodes [37]. Only one root node can exist in one DT. The other nodes are either leaf node or a non-leaf node. DT makes decision using decision node, which splits the data into its desired class. There are lots of possibilities to measure the split of the subtrees. Since it is a top-down approach, a small mistake made at the beginning can largely impact the performance since each iteration is based on the first.

Ensemble Approach based Classifiers:

The logic behind ensemble models is that it’s better to use many mediocre models than one good model. It combines many single models and creates an ensemble model. Examples of ensemble model is Random Forest (RF), AdaBoost [38].

The target of this study is to detect whether a job post is fraudulent or not. Identifying and eliminating these fake job advertisements will help the job seekers to concentrate on legitimate job posts only. In this context, a dataset from Kaggle is employed that provides information regarding a job that may or may not be suspicious. Before they fit this data to a model for training, they did some pre-processing to dataset. They remove null values, stop-words, irrelevant attributes are also removed and extra space removal. With this cleaned data they can obtain feature vectors. They have used Naive Bayes Classifier, Decision Tree Classifier, Multi-Layer Perceptron Classifier, K-nearest Neighbor Classifier,

AdaBoost Classifier, Gradient Boost Classifier and Random Tree Classifier for classifying job post as fake or not.

To maximize the performance, they are not using default parameters. Then training the model takes place. After these tests are conducted. For ensemble models it is observed that Random Forest performance is better. RF gives an accuracy of 98.27%, Cohen-kappa score as 0.74, F1-score 0.97, MSE 0.02. Therefore, they have concluded that Random Forest is the best candidate for this fake job prediction.

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN

The arrangement of software components on devices is referred to as software system architecture. Two components that are closely connected can also be co-located or deployed on distinct machines. The placement of the software components will also have an impact on the performance and dependability of the system. The purpose of having a system architecture is to design comprehensive solutions centered on logically connected and consistent principles, ideas, and attributes. Software architecture is a type of system blueprint that is essential for understanding, negotiation, and communication among all stakeholders (users, customers, management, etc.). It makes the entire system easier to grasp as well as the decision-making processes very efficient. The process of developing the framework, product innovation, components, protocols, and information for a system in order to meet defined criteria is known as systems design. Systems design may be described as the application of systems approach to the creation of products. When a system can fulfil the needs of the end user, it is considered reliable.

When building a system, we might as well have prepared to implement a selection of characteristics and services. If the platform can satisfy all such features without having to wear out then the structure can be deemed reliable. A fault-tolerant system is something that can continue operating reliably in the event of problems. Faults are mistakes that occur in a specific element of the system. The occurrence of a malfunction does not ensure that the system will fail. Failure occurs when a system is unable to operate as intended. It can no longer deliver certain services to end consumers. Availability is a feature of a system that seeks to maintain an agreed-upon level of organizational performance, often known as uptime. In order to service the user's requests, a system must offer high availability.

The objective or appearance of a system architecture and design requirements specification may differ depending on the project, but they all serve the same fundamental function. That is, to guarantee that the software's owners and developers have a solid awareness of its future specifics and project estimates. Project estimations are a crucial aspect of project planning that include cost estimates, resource allocation, and project length. Knowing what technological requirements your program

requires might assist you in determining these parameters more precisely.

3.1 EXISTING SYSTEMS

Table 3.1 Existing Systems Comparisons

Methodology	Metrics	Drawbacks
Naïve Bayes Classifier	F1 score – 0.72 MSE – 0.52	Naive Bayes assumes that all predictors are independent, rarely happening in real life.
AdaBoost Classifier	F1 score – 0.98 MSE – 0.03	It needs a quality dataset. Noisy data and outliers have to be avoided before adopting an Adaboost algorithm.
K-Nearest Neighbours Classifier	F1 score – 0.96 MSE – 0.04	Require high memory – need to store all of the training data. Sensitive to the scale of the data and irrelevant features.
Multi-Layer Perceptron Classifier	F1 score – 0.96 MSE – 0.05	MLP requires tuning several hyperparameters such as the number of hidden neurons, layers, and iterations.
Decision Tree Classifier	F1 score – 0.97 MSE – 0.03	A small change in the data can cause a large change in the structure of the decision tree causing instability.

3.1.1 CHALLENGES IN THE EXISTING SYSTEM

Naive Bayes Classifier assumes that all predictors are independent, rarely happening in real life. Adaboost Classifier needs a quality dataset. Noisy data and outliers have to be avoided before adopting an Adaboost algorithm. K-Nearest Neighbour Classifier Require high memory – need to store all of the training data. Sensitive to the scale of the data and irrelevant features. Multi-Layer Perceptron Classifier requires tuning a number of hyperparameters such as the number of hidden neurons, layers, and iterations. Decision Tree Classifier - A small change in the data can cause a large change in the structure of the decision tree causing instability.

3.2 PROJECT REQUIREMENTS

Requirements are a description as to what should be done. They are statements about how the system should function, or about a system property or characteristic. When requirements are ambiguous, initiatives run the risk of failing to provide what is required. Missed criteria, at the very least, necessitate rework. There will very certainly be negative consequences for both the schedule and the budget.

3.2.1 FUNCTIONAL REQUIREMENTS

Functional requirements are requirements that characterize the product's behavior. These requirements would be the requirements that the end user requests for as the basic facilities for the system to provide. As part of the contract, all of these features must be included into the system. These are depicted or stated in the form of input to the system, operation executed, and intended outcome. In contrast to non-functional needs, they are essentially the user-specified criteria that can be seen immediately in the finished product. This part of the document talks about the developmental potential of the software product's needs. It is the initial phase in the process of requirements analysis. It enumerates the needs of a certain software system. The specific libraries such as sk-learn, Pandas, NumPy, matplotlib, and seaborn are detailed below.

3.2.2 NON – FUNCTIONAL REQUIREMENTS

Explains the ambient conditions or attributes necessary for such a product to function properly. These are the quality restrictions that the software must meet in accordance with the project contract. The significance or depth to which these aspects are incorporated varies depending on the project. Non-behavioral requirements are another name for them. A non-functional requirement establishes a software system's quality attribute. It restricts how the software system satisfies the functional requirements. Assists you in testing the software's performance. Non-functional requirements – describe the environmental circumstances or characteristics required for the product to perform properly. An example would be equipping the model to possess time constraints to process every request that comes in to be done within 10 seconds. This also includes Testing such as Performance, Stress, Usability, Security testing, etc. that are performed on the model to ensure the absolute best for the users. Thereby, allowing them to have a very smooth experience and leave them satisfied with the results of their online job search.

3.2.3 TECHNICAL REQUIREMENTS

When working on a project or developing software, technical requirements explain the technical features and difficulties that must be addressed in order for the project or product to function and execute properly. These technological characteristics might allude to issues such as performance difficulties, software dependability, and ease of access. Technological requirements, in essence, serve

as solutions to technical problems. Technical requirements are crucial because they define how software should perform and behave. This assist both developers and users in determining the optimal method to utilize the product. A document with properly specified specs aids in the creation of a project or software with a good implementation procedure. This is referred to as technical requirement.

TO BUILD THE MODEL

Software Requirements:

- Python 3.7.x
- Anaconda/Google Colab/VSCode (Any code editor with Python support)
- Windows/Linux/MacOS (Any OS that supports the above requirements)

Hardware Requirements:

- Any modern Dual Core CPU (Intel Celeron+/ AMD Athelon+)
- 4GB+ RAM
- 1GB HDD space

TO DEPLOY AND RUN THE MODEL

Software Requirements:

Python 3.7.x

Flask

VSCode/Sublime Text (Any Text Editor of your choice)

Hardware Requirements:

- Any modern Dual Core CPU (Intel Celeron+/ AMD Athelon+)
- 4GB+ RAM
- 1GB HDD space

3.3 PROPOSED SYSTEM

The proposed method is to build a machine learning model to classify the real or fake job posting to overcome this method to implement a machine learning approach. The proposed method is to build a machine learning model to classify a real or fake job posting. Many systems use neural networks to

predict such cases, but if the data has any noise or error, there is a possibility of overfit. So, the data must be as clean as possible, which is difficult in practical applications. Logistic Regression and Random Forest Classifier are less complex than Neural Network based models hence they are less susceptible to overfitting. LSTM while having better memory retention than RNN, it is extremely inefficient and requires a lot of time for processing. Random Forest Classifier is much more time efficient. In Random Forest Classifier, we don't have to clean the training data as much as RNN. The logic is that an ensemble of many mediocre models would still fare better than a single good model. Because of this, Random Forest Classifier is less prone to overfit. The dataset is first preprocessed and the columns are analyzed to see the dependent and independent variables then different machine learning algorithms would be applied to extract patterns and to obtain results with maximum accuracy.

3.4 DESIGN OVERVIEW

In design overview, we go through each part of the algorithm, here we briefly go through data preprocessing, EDA, comparison, training the model, tuning and testing.

3.4.1 DATA PRE – PROCESSING

The raw data is examined for quality, cleaned, converted, and reduced to a comprehensible format. It is divided into four primary steps: data cleansing, integration, reductions, and transformation. This is where NLP will enter the picture. Before modelling, the data is cleaned by eliminating punctuation, stop words, and digits, which do not really provide information about the target. We eliminate stop words that feature articles such as an, a, and the from every text in the dataset, leaving only the keywords for the analysis step. For the text characteristics, we calculate the difference between the mean number of words in fake and real job listings. The total number of words for the company profile and requirements is then determined.

3.4.2 EDA

Exploratory Data Analysis also known as 'EDA' is a critical stage in identifying trends, patterns, insights and anomalies in a dataset and create a hypothesis based on our current understanding of it. EDA is not a formal procedure with a set of rules to follow. EDA is, above all, a state of mind. During the early stages of EDA, now you just should feel comfortable to study any notion that comes to mind. A few of these concepts will come to fruition, while others will fail. As you continue to explore, you'll come across a few really fruitful regions that you'll ultimately write up and transmit to others. Even if

the queries are handed to you on a silver platter, EDA is an important part of any data analysis since data quality must be regularly assessed. Data cleansing is just one application of EDA: you ask whether your data meets your needs. Through exploratory data analysis, we identified which of the 18 features were particularly frequent among fake job listings. We use a pie chart to analyze the percentage distribution of each feature with the fraudulent output. Using this strategy, we discovered that the majority of the fraudulent job postings lacked a firm profile as well as valid job descriptions and requirements. The model is trained by passing the input data through the algorithm and comparing the processed output to the sample output. The correlation finding is used to adjust the model.

3.4.3 COMPARING THE FOUR MODELS

Four machine learning classification models - Random Forest Classifier, Logistic Regression, XGBoost, and Support Vector Machines are separately assessed by evaluating them against four unique EMSCAD dataset features: company profile, job description, benefits, and requirements. These characteristics were chosen specifically because they identify real job postings from fake ones.

3.4.4 TRAINING

The cleaned dataset is split into separate training and testing datasets in such a way that the number of real and fraudulent jobs are balanced. The 5 models are then trained to differentiate and identify the nature of the jobs.

3.4.5 TESTING AND PERFORMANCE MEASUREMENTS

The 4 ML models and the Ensemble model are tested against four features of the dataset based on which their performances are evaluated by 3 metrics: Precision and Recall which is then used to calculate the F1-score that determines the accuracy of the model created.

3.4.6 TUNING THE MODEL

We must now tweak the model that we have developed. The practice of providing weights for every model is known as tuning. As a result, the accuracy of each model's output is weighted differently. This manner, we may take advantage of the benefits of each model while without detracting from the efficiency of the ensemble classifier. We use the brute force strategy, assigning different weights to every classifier and then evaluating their performance; the model achieves the best results with weights of 10,10,10,10 to every classifier and a threshold of 20.

3.4.7 DEPLOYMENT

The Ensemble Model thus created is deployed and connected to a Flask application which is the interface the user will interact with and enter details which the model will analyze such as the company's profile, the job description, benefits and requirements.

3.5 SYSTEM EVOLUTION DESCRIPTION

We research and test various Machine Learning Classification models and came to the following conclusions:

- Naive Bayes - It's frequently used to build machine learning models that can make quick predictions. A probabilistic classifier predicts based on the likelihood of an object. Assumes that all predictors are independent, which seldom occurs in practice. It has an issue with zero-frequency.
- Adaboost Classifier - To improve a poor classifier that is marginally better than a random guess to a good classifier. It is known as Adaptive Boosting because the weights are reassigned to each instance, with larger weights allocated to erroneously assigned instances. Needs a quality dataset. Noisy data and outliers have to be avoided before adopting an Adaboost algorithm.
- K-Nearest Neighbor Classifiers - A supervised machine learning method that can address classification as well as regression issues. To fill up missing values and resample datasets. Requires high memory and needs to store all of the training data.
- Multi-Layer Perceptron Classifier - A feedforward artificial neural network produces a set of outputs based on a set of inputs. It is distinguished by many layers of input nodes that are connected in a directed graph between the input and output layers. A number of hyperparameters, such as the number of hidden neurons, layers, and iterations, must be tuned.
- Decision Tree Classifier - To successfully handle non-linear datasets, it generates the categorization model by constructing a decision tree. It manages all forms of data well. A small change in the data can cause a large change in the structure of the decision tree causing instability.

3.6 SYSTEM ARCHITECTURE DIAGRAMS

An architectural diagram is a graphical representation of how software system components are physically implemented. It displays the software application's overarching architecture, as well as the relationships, constraints, and boundaries that exist between each component.

3.6.1 ARCHITECTURE DIAGRAM

The architecture diagram is a simple and clear representation of how the model functions to process the data, create, train, test and tune the model that will detect and differentiate fraudulent and real job postings to help users avoid getting scammed while searching and applying for jobs online. The essential feature of such a diagram would be that it arranges and clearly depicts the system's users and high-level dependencies. It merely takes a couple of minutes to draw the diagram when the concept and work is completed. This diagram is an important step for app and software developers to show the basic layout of the system by splitting functional regions into strata. It demonstrates how a normal software system could interact with its users, other computers, information sources, and services.

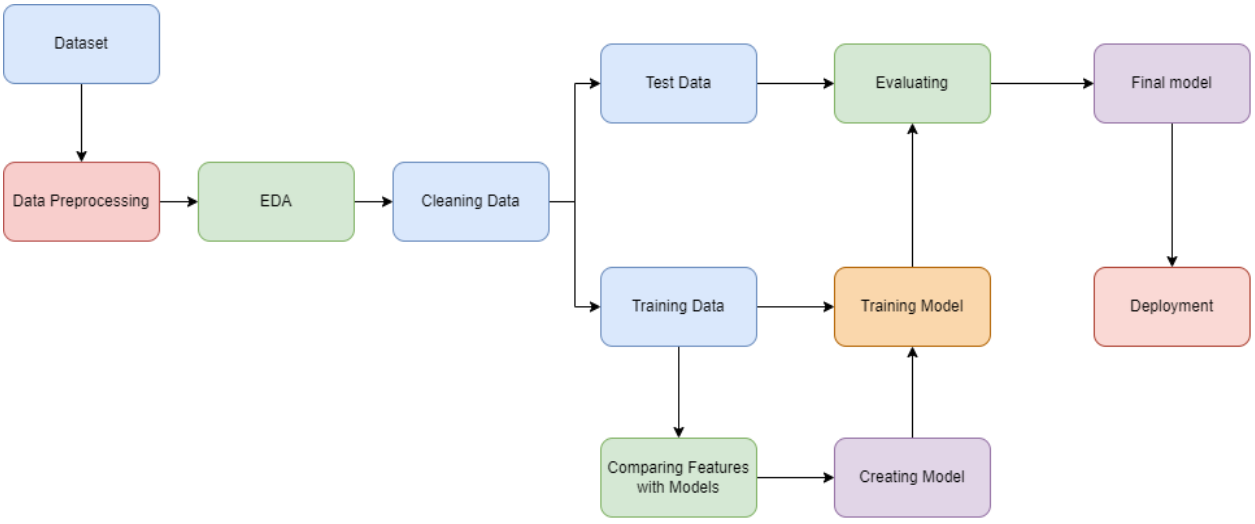

Fig 3.1 System Architecture

3.6.2 SEQUENCE DIAGRAM

The sequence diagram gives a process view of the model used and is presented in the form that is similar to a flow chart diagram. It describes the sequence of data and the communication between the input and outputs of the system components. It reflects the functional flow of the system and clearly

depicts the flow of data within the system for easy understanding of the model and its internal working. The dataset is first loaded into the code editor and thoroughly analyzed to see if it is a balanced one or not. It is then checked for recurrent patterns, checking if there is any missing data, values, garbage values, stop words etc. The dataset undergoes pre – processing and cleaning it is saved as a new dataset which is again loaded into the editor. Now, comes the part where we play with various machine learning models using our dataset and its features. Each of the models are tested, one by one and their performance results are saved. As the next step, we decide to make a combination of all the models we just tested in order to obtain an ensemble model capable of making highly accurate predictions to help protect and inform users of fraudulent job postings online and allow users to find only the real and authentic job listings. This process is very hassle free as all it requires is for the user to enter just four details regarding the job listing, they want to check. First, enter the name of the company, the description they have provided, the qualifications for the position they are recruiting for and the benefits the company provides for the hires.

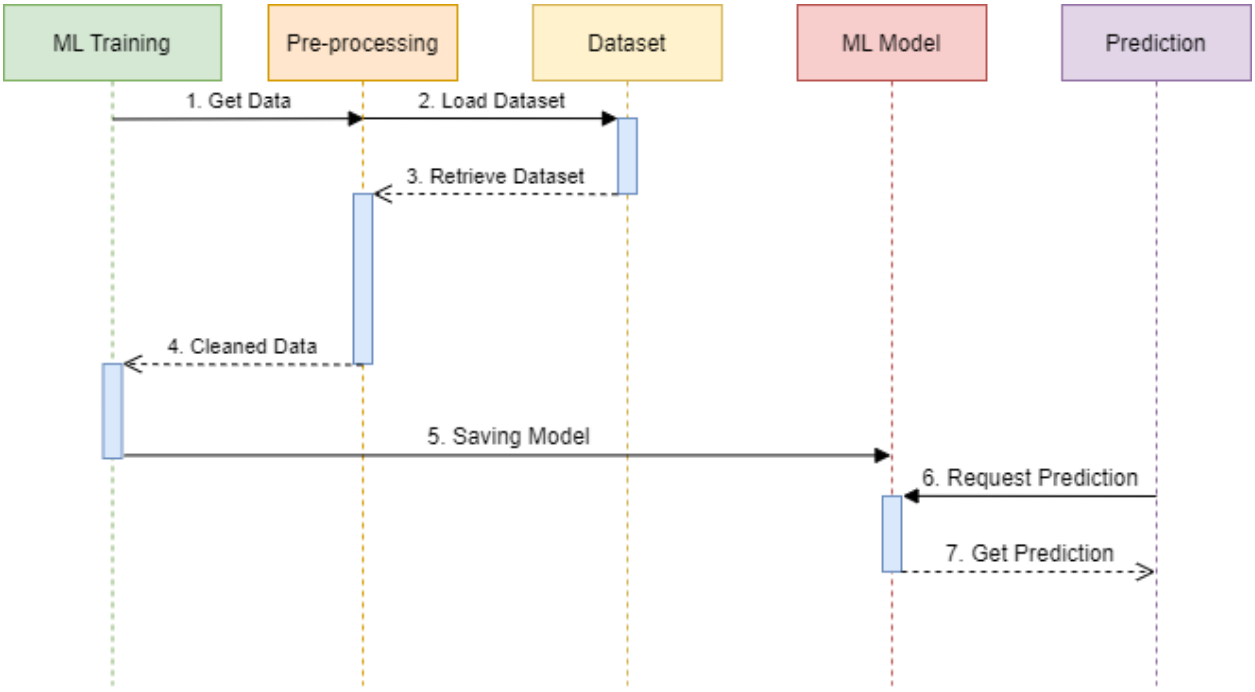

Fig 3.2 Sequence Diagram

3.6.3 ACTIVITY DIAGRAM

An activity diagram, similar to a data flow diagram, visually depicts a series of actions or the flow of power in a system. In business process modelling, activity diagrams are frequently employed. They

may also use a use case graphic to illustrate the stages. Modeled activities can be both sequential and concurrent. The activity diagram aids in visualizing the flow of work from one action to the next. It emphasized the flow state and the sequence in which it happens. The flow can be in a sequence, branching, or concurrent, and it has forked, joined, and so on to cope with such flows. An object-oriented flowchart is another name for it. It includes activities that are made up of a series of actions or processes that are used to model the behavioral diagram.

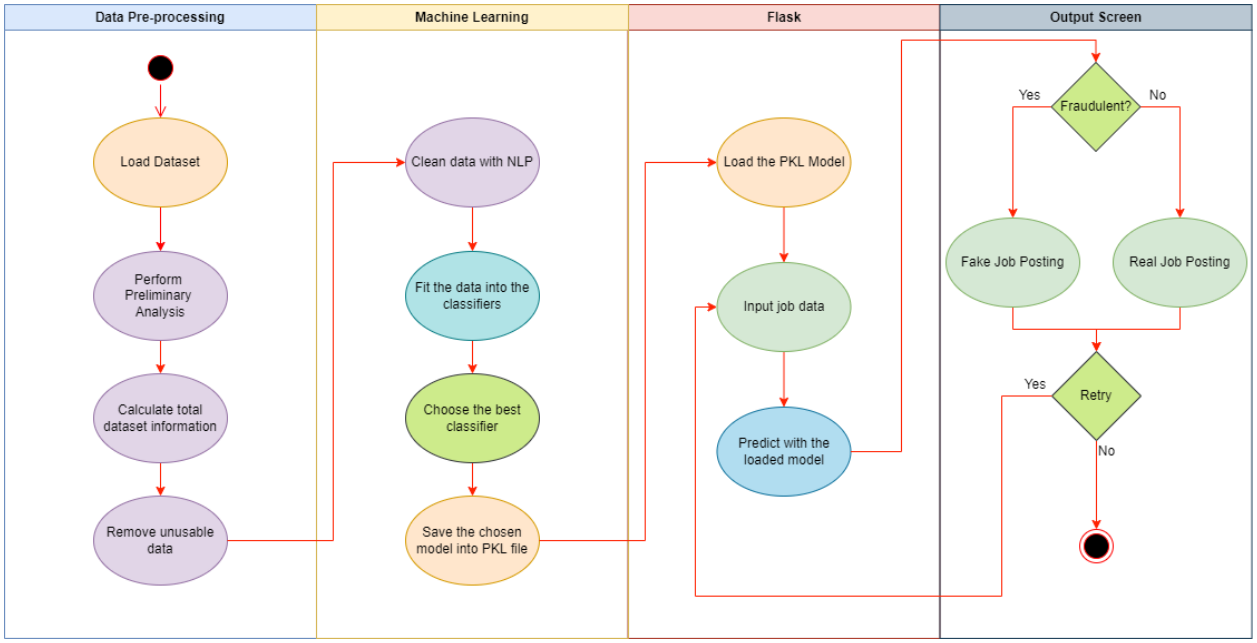

Fig 3.3 Activity Diagram

CHAPTER 4

METHODOLOGY

The dataset is cleaned by removing stop words, punctuations etc. From the exploratory data analysis, the distribution of real and fraudulent jobs is imbalanced. The dataset is split into train and test sets such that the datasets are balanced to avoid bias towards any of the classes. From the dataset, four distinct features are taken – Job Description, Job Requirements, Job Benefits and Company Profile and first, individually tested with the Random Forest Classifier, Logistic Regression, Support Vector Machines and XGBoost models. The individual performances of each model against each feature were determined by calculating their F1 Scores from Precision and Recall metrics of each feature for their respective models. To obtain a model that can make predictions with the highest accuracy, the 4 models tested are combined to create our very own Ensemble model.

4.1 OBJECTIVE

The goal is to develop a machine learning model that can classify real or fake jobs. We create an ensemble model with Random Forest Classifier, Logistic Regression, SVM, and XGBoost for each feature. For each feature, a classifier is chosen in such a way that it gives maximum accuracy. The final model is implemented through Flask.

4.2 PROJECT GOALS

- Exploration data analysis of variable identification
    - Loading the given dataset
    - Import required libraries packages
    - Analyse the general properties
    - Find duplicate and missing values
    - Checking unique and count values
- Uni-variate data analysis
    - Rename, add data, and drop the data
    - To specify data type
- Exploratory data analysis of bivariate and multivariate

    - Plot diagram of pair plot, heatmap, bar chart and Histogram
- Method of Outlier detection with feature engineering
    - Pre-processing the given dataset
    - Splitting the test and training dataset
    - Comparing the Decision tree and Logistic regression model and random forest, etc.
- Comparing algorithms to predict the result
    - Based on the best accuracy

4.3 SCOPE:

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

4.4 TECHNICAL MODULES

We have mentioned the modules that our project has. This gives a brief description of all the modules and explains in detail.

4.4.1. DATA PREPROCESSING

Dataset:

The dataset contains 17880 entries with 18 features. The dataset is sent for the cleaning process. Fig 4.1 lists all the features available in the dataset. Fig 4.2 mentions the data distribution between the two classes before data is being cleaned.

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

Fig 4.1 List of features in dataset

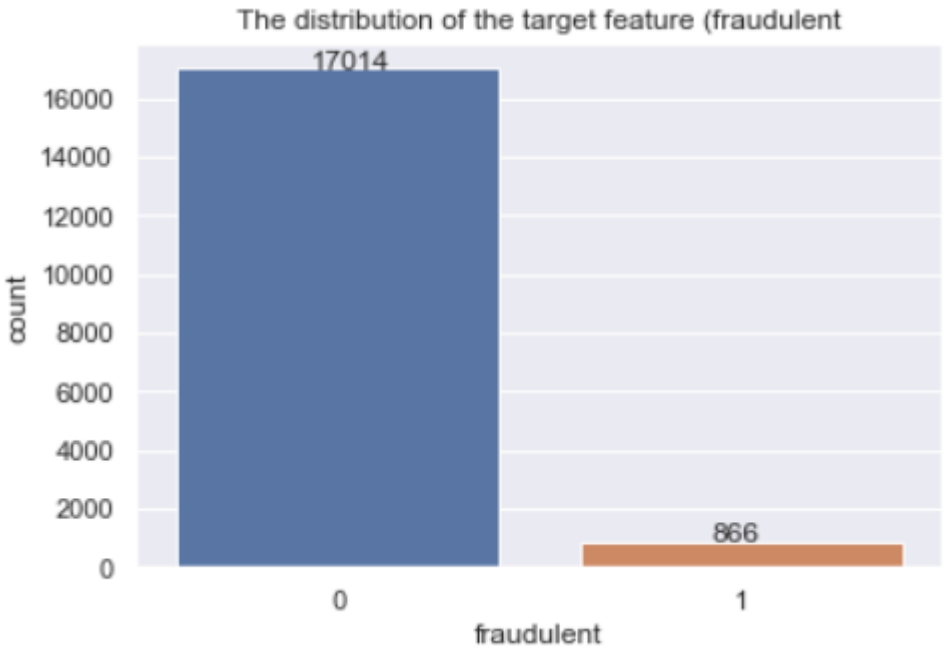

Fig 4.2 Distribution of dataset for fraudulent and non-fraudulent before cleaning

Cleaning:

We first separate all the features into 4 parts.

- **Binary type:**

  Features like telecommuting, has_company_logo, and has_questions are binary types, with only two answers, true is represented as numeric 1 and false is represented as numeric 0.

- **Category type:**

  Features like department, industry, function, employment_type, required_experience, and required_education are bounded by certain categories.

  - Department has a wide range like Marketing, Sales, R&D, Production, etc
  - Industry has Computer Software, Hospital & Health Care, Information Technology and Services, Management Consulting, etc.
  - The function has Customer Service, Management, Information Technology, etc.
  - Employment_type has Full-time, Contract, Part-time, Temporary, etc.
  - Required_experience has Mid-Senior level, Associate, Associate, etc.
  - Required_education has bachelor’s degree, High School or equivalent, Master's Degree, etc.

- Text type:

The features that have text as data are known as Text features. Features like title, company_profile, description, requirements, and benefits, all describe the job posting, and it's an important component for identifying job postings. This is where we also are using NLP.

- Complex type:
  These are certain types of features that don't fall under any definite category. They are features like location and salary_range.

Feature Preparation:

This is the part where we take each type of data we separated in the previous phase and clean them

Text Type:

Adding indicators

Fill-NA: First go through the dataset and fill the empty inputs with an empty string.

Pre-process text: This is where we will be using NLP. For each text data, we remove stop words. Stop words contain articles like “an, a, the”, Etc. It removes unimportant words and keeps the keywords for analysing phase. We create another column named “company profile specified”. This has a Boolean value and tells whether there is a company profile or not. Likewise, we create column names as “description specified”, “requirements specified”, and “benefits specified”.

Complex Type:

Complex type consists of location and salary, so we work on both separately

Location

Each location is mostly of the type of city, state, country. All are separated by commas (,). So, we split these entities.

Sometimes there won't be 3 entities alone, it can be two or four or five, or in some cases, 3 entities might be there, but they would have not mentioned some entity and rather mentioned something else. In those cases, we attach the string “Unspecified”

After we split all data to country, stale, and city. We create 3 columns in the table namely country, state, and city. Then we add all data we have collected to all columns.

Finally, we remove the column location since we put it all into 3 rows as shown in Fig 4.3.

	country	state	city
0	US	NY	New York
1	NZ	Unspecified	Auckland
2	US	IA	Wever
3	US	DC	Washington
4	US	FL	Fort Worth
5	US	MD	Unspecified
6	DE	BE	Berlin
7	US	CA	San Francisco

Fig 4.3 New columns are created for location

Salary range

Salary range is in the format min salary - max salary.

If the range is not specified, we replace the null value with a range 0-0 (zero to zero).

Then just like in the location column, we split the min salary and max salary

Then we create two columns called min_salary and max_salary, and we add the values to these columns like shown in Fig 4.4.

	min_salary	max_salary
6	20000	28000
7	0	0
8	0	0
9	0	0
10	100000	120000

Fig 4.4 New columns for salary range

We also created another column called salary_specified, this is for reference, it has a Boolean value like shown in Fig 4.5.

	min_salary	max_salary	salary_specified
6	20000	28000	1
7	0	0	0
8	0	0	0
9	0	0	0
10	100000	120000	1

Fig 4.5 salary_specified column

Finally, we drop the salary_range column from the table. We check if there are any null data with the features. For Boolean type naturally, there is no null value. In category type, there are null values, so we can fill all null values with the string “Unspecified”.

With this, all values are filled, and no null values are present.

Finally, for the fraudulent column, it's a Boolean type that says whether the job posting is fake or real. We change the Boolean to a text saying whether the job posting is real or fake.

We save the cleaned data to “cleaned-data.csv”

From Fig 4.6 we can see that after data pre-processed the dataset is balanced.

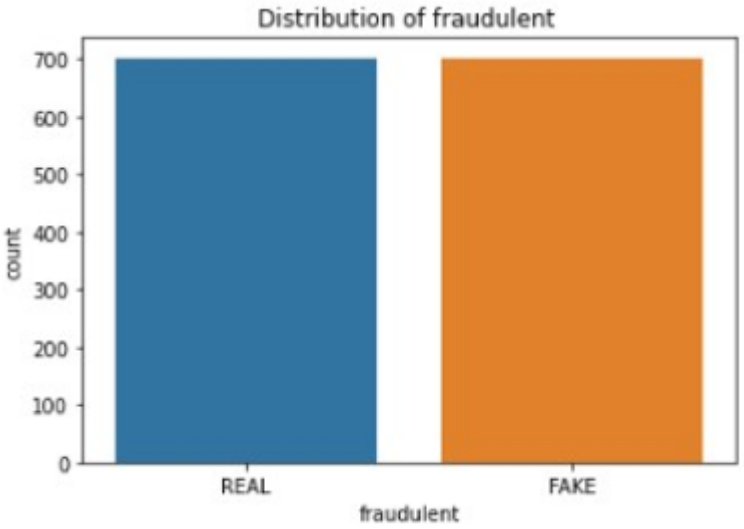

Fig 4.6 Balanced Data set after cleaning

4.4.2. EXPLORATORY DATA ANALYSIS

In modern statistics and machine learning, data visualisation is a crucial ability. Statistics is concerned with quantitative data descriptions and estimations. Data visualization is a valuable set of tools for acquiring a qualitative understanding of data. This might be useful for spotting trends, faulty data, outliers, and other things when studying and getting a dataset. Data visualizations may be used to communicate and show crucial relations in graphs and charts that are far more visceral and meaningful to stakeholders than measurements of correlation or importance with a little subject expertise. Data visualization and data exploration are areas in and of themselves, and it will be recommended that you read some of the books indicated at the conclusion for further information.

Data may not make sense unless it is presented in a visual format, such as charts and graphs. The ability to see data samples and other objects rapidly is a crucial talent in both applied statistics and applied machine learning. It will show you how to utilize the many sorts of plots available when visualizing data in Python to help comprehend your own data.

4.4.3. CORRELATION

For every feature, we compare it with the fraudulent output, and see how much correlation is there between each feature with the output.

4.4.4. COMPARISON

When we have a fresh dataset, it's a smart option to display it using a variety of ways so we can see it from multiple angles. Model selection follows the same logic. To pick the one or two to complete, we should look at the direct results of your machine learning techniques in a variety of methods. Using various visualization approaches to display the average accuracy, variance, and other features of the distribution among model accuracies is one way to do this.

In the next part, we see how to accomplish it in Python. The key to a true comparison of ML algorithms is to ensure that each method is assessed in the very same way on the very same data, which may be accomplished by requiring each method to be tested on the same test harness.

The following algorithms were compared with the 4 features that is compatible with the output

- Random Forest Classifier
- Logistic regression
- SVM
- XGBoost

Each method is evaluated using the K-fold cross validation technique, which is set up with the same random seed to ensure that the training data is divided in the same way and that each strategy is evaluated in the same way. Separate the training and testing sets. It is feasible to predict the outcome by comparing accuracy.

Random Forest Classifier:

RFC is just a combination of many decision trees. A Decision tree is, as the name suggests, takes decisions at times based on the data that are present. RCF is an ensemble model, an ensemble means a combination of many decision trees. With the data collected, the decision tree gives output and with

the output, we get from the decision tree, we create another decision tree for the outputs we get to get the accurate output while also considering many factors at the same time. The logic behind this is that a combination of many mediocre models is better than one good model. We need a Random Forest because we need features that have some predictive power. The trees of the forest and their predictions need to be uncorrelated so that the output does not waver. Some Machine Learning models need data in a specific format, such as the Random Forest method, which does not accept null values. As a result, null values must be handled from the initial source data set in order to run the random forest method. Another consideration is that the data set be written in such a manner that many Machine Learning and Deep Learning algorithms may be run on the same dataset.

Logistic regression:

Let us consider a scenario where we must tell whether that's a spam email or not. If we use linear regression for this, then there is a need for setting up thresholds and that threshold-based classification is made. LR is mainly used when the target is categorical. In our project, we just must mention whether the job posting is fake or not, which falls into only two categories. It is a process of modelling the probability of a certain outcome from the input. As we mentioned before, LR works well with binary outcomes. Of Course, there are Multinomial logistic regression where the outcome can be more than two. It is considered as a supervised ML algorithm that is useful for binary classification problem. The difference between linear regression and logistic regression is that in logistic regression, the outcomes are bound between 0 and 1.

It is said that Logistic regression is a transformation of linear regression using a function called sigmoid function. The advantage is that it can be used both for classification and also class probability estimation because it is tied with logistic data distribution.

Support Vector Machines

Support vector machines are a type of machine learning technique that can be used for both regression and classification, however it is more commonly employed for classification. SVM is an n-dimensional space model. We plot the data items in any nth dimension, where n is the number of features we have. As a result, the more features we have, the more dimensions we can deal with. The classification is then carried out by locating the hyperplane that separates these two classes.

The main advantage is that it can provide high-dimensional spaces. It is flexible when the number of dimensions is greater than the number of samples that we are provided. The reason we are implementing this classifier is that it can still predict the output with some missing samples as it was mentioned before.

XGBoost:

XGBoost is an optimized gradient boosting library that is designed to be efficient, flexible, and portable. From the Gradient Boosting framework, it can implement the machine learning algorithms. The reason to use XGBoost is because it has faster execution speed when compared to other gradient boosters. The individual model performance given by XGBoost is also incredible.

It is an open-source implementation of gradient boosted tree techniques that is popular and efficient. Gradient boosting is supervised learning that combines an ensemble of estimates from a set of simple and weak models to try to properly predict a target variable. As a result, because our dataset is severely uneven, XGBoost can be of great assistance to our system.

4.4.5. TRAINING

The cleaned dataset is split into separate training and testing datasets in such a way that the number of real and fraudulent jobs are balanced. The ensemble model is then trained to differentiate and identify the nature of the jobs.

4.4.6. TUNING THE MODEL

Now that the model is created, we must tune it. Tuning is the process of assigning weights to each model. So that the result of each model is given separate importance based on their accuracy. This way, we can get the advantages of each model and it will not pull down the performance of the ensemble model. We are using brute force method, we assign different weights to each classifier and then check their performance, the highest performance that is achieved by the model is of by the weights 10,10, 10, 10 for each classifier and threshold of 20.

4.4.7. TESTING & PERFORMANCE MEASUREMENTS

The 4 ML models and the Ensemble model are tested against four features of the dataset based on which their performances are evaluated by 3 metrics: Precision and Recall which is then used to calculate the F1-score that determines the accuracy of the model created.

Measurement Metrics:

Accuracy:

Logistic regression algorithm also uses a linear equation with independent predictors to predict a value. The predicted value can be anywhere between negative infinity to positive infinity. It needs the output of the algorithm to be classified variable data. Higher accuracy predicting the result is a logistic regression model by comparing the best accuracy.

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

$$\text{Accuracy} = \frac{(tp+tn)}{(\text{tp+tn+fp+fn})}$$

tp - number of true positives
fp - number of false positives
fn - number of false negatives
tn - number of true negatives

Precision and Recall:

Both are used for evaluating models of a particular class of interest, also known as the positive class.

Precision is, for all positive predictions, how many are real positives?
Recall is, for all real positive cases, how many are predicted positive?

Formula:

$$\text{Precision} = \frac{tp}{(tp+fp)}$$

$$\text{Recall} = \frac{tp}{(tp+fn)}$$

tp - number of true positives

fp - number of false positives

fn - number of false negatives

F1 Score:

It is used to assess the correctness of a model on a dataset. It assesses binary classification systems that assign positive or negative labels to examples.

F1-Score Formula:

$$F1 = \frac{2}{\frac{1}{recall} \times \frac{1}{precesion}}$$

$$= 2 \times \frac{precision \times recall}{precision + recall}$$

$$F1 = \frac{tp}{tp + \frac{1}{2}(fp+fn)}$$

tp - number of true positives

fp - number of false positives

fn - number of false negatives

False Positives (FP):

When the actual class is negative, and the projected class is positive. For example, if the real class states this passenger died, but the anticipated class predicts this passenger will live.

False Negatives (FN):

When the actual class is positive, and the projected class is negative. For example, if the real class states this passenger survived, but the anticipated class predicts this passenger died.

True Positives (TP):

These are the accurately predicted positive values, indicating that the actual class value is yes, and the projected class value is also yes. For instance, if the actual class value indicates that this passenger survived, and the projected class also suggests that this passenger survived.

True Negatives (TN):

These are the accurately predicted negative values, indicating that the actual class value is no, and the projected class value is also no. For instance, if the actual class value indicates that this passenger died, and the projected class also suggests that this passenger died.

Mean Squared Error:

This is the most basic loss function. So basically, we subtract the model's predictions and the truth, square it, and take the average across the entire dataset.

Formula:

MSE = $\frac{1}{N}\sum_{i=1}^{N}(y_{i-}\hat{y_i})^2$

N - The number of samples we are testing

y_i - Ground truth

$\hat{y_i}$- Model's prediction

MSE and F1-score and indirectly proportional, the more MSE is, the less F1-score is. So, our goal is to minimize as much MSE as possible. MSE can never be negative.

4.4.8. DEPLOYMENT

The Ensemble Model thus created is deployed and connected to a Flask application which is the interface the user will interact with and enter details which the model will analyze such as the company's profile, the job description, benefits, and requirements.

CHAPTER 5

CODING AND TESTING

5.1 CLEANING THE DATASET

There will always be some amount of inaccuracy with datasets regardless of how the data in it is obtained. What we call, "Messy data" is data that is filled with irregularities and anomalies. Though some of the variations are real since they represent variance in the environment, others are most likely due to measurement, input, processing or data integration errors. These might range from errors human negligence, badly designed records, or merely an inability to control the format and the kind of data acquired from various external data sources. Such inconsistencies cause chaos when attempting to analyze data. Prior to actually performing data pre - processing for analysis, effort should be made to ensure that the data is as reliable and precise as feasible.

5.1.1 IMPORTING LIBRARIES

Python's import function is analogous to header files in other languages such as C and C++. Python modules can access code from other modules by importing the specific file or method using import. The import function is the most used method of triggering the import mechanism in python. Python considers a file to be a module. The module must be integrated using the import keyword before it can be used. By importing the module, the methods and variables included within the file may be utilized in another program. This capability is accessible in other programming languages as well.

```
import pandas as pd
import numpy as np
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
```

Here we import the required libraries for the model Pandas is a free and open source data analysis package built on Python programming language. The pd alias is often used to import this library. alias: In Python, an alias is a different name for the same item. Instead of pandas, the Pandas package is now referred to as pd. The as pd section of the code then instructs Python to assign pandas the alias pd. The pandas functions can be used by having to type pd.function name instead of pandas.function name.

The import NumPy line of code instructs Python to import the NumPy library into the current environment. The as np section of the code then instructs Python to assign NumPy the alias np. You may utilize NumPy functions by just entering np. NLTK is a Python toolkit for working with natural language processing (NLP). It provides us with a large number of test datasets for various text processing libraries. NLTK may be used to execute a number of tasks such as tokenizing, parse tree visualization, and so on. Import stop words from nltk.corpus. This is a list of English lexical stop words. That is, most NLP actions, such as part-of-speech labelling, tokenization, and parsing, ignore these words. Tokenizers are program that convert strings into lists of substrings. Tokenizers, for example, can be used to locate letters and commas, whitespace, full stops etc. in a string.

5.1.2 READING AND ANALYZING THE DATASET

```
data = pd.read_csv("./fake_job_postings.csv")
data.head(10)
```

5.1.3 SPLITTING THE DATA INTO GROUPS OF FEATURE TYPES

```
bin_features = ['telecommuting', 'has_company_logo', 'has_questions']
cat_features = ['department', 'employment_type', 'required_experience',
                'required_education', 'industry', 'function']

text_features = ['title', 'company_profile', 'description', 'requirements',
'benefits']
complex_features = ['location', 'salary_range']
```

The features of the dataset are split based on their respective feature types. Here, telecommuting, company logo, etc. all belong to ‘bin_features’ as they are binary features. Features like department, employment type, experience, educational qualifications etc. are categorical features and features like the company’s title, profile, description, requirements and benefits fall under the text features category. However, certain features like location and salary range are considered complex features here.

```
data.drop('job_id', axis=1, inplace=True)
```

As the feature, ‘job_id’ is of no relevance to our model and usage, we drop the entire column.

5.1.4 ADDING INDICATORS AND FILLING NA

```
for feature_name in text_features[1:]:
    un_feature_name = f"{feature_name}_specified"
```

```
    data[un_feature_name] = (~data[feature_name].isna()).astype('int')
    bin_features += [un_feature_name]
data.head()[text_features + bin_features[-4:]]
```

```
for feature_name in text_features[1:]:
    data[feature_name].fillna('', inplace=True)
nltk.download('stopwords')
nltk.download('punkt')
nltk_lang = stopwords.fileids()
stop_words = set(stopwords.words(nltk_lang))
porter = PorterStemmer()
def preprocesstexts(texts):
    preprocess_texts = []
    for text in texts:
        text = ''.join([sym.lower() for sym in text if sym.isalpha() or sym == ' '])
        tokenized_text = word_tokenize(text)
        tokenized_text_wo_sw = [word for word in tokenized_text if word not in
stop_words]
        tokenized_text_wo_sw_stem = [porter.stem(word) for word in
tokenized_text_wo_sw]
        preprocess_texts += [' '.join(tokenized_text_wo_sw)]
    return preprocess_texts
for feature_name in text_features:
    data[feature_name] = preprocesstexts(data[feature_name])

data[text_features].head()
```

company_profile	description	requirements	benefits
food weve created groundbreaking awardwinning ...	food fastgrowing james beard awardwinning onli...	experience content management systems major pl...	
seconds worlds cloud video production service ...	organised focused vibrant awesomedo passion cu...	expect youyour key responsibility communicate ...	get usthrough part seconds team gainexperience...
valor services provides workforce solutions me...	client located houston actively seeking experi...	implement precommissioning commissioning proce...	

Fig. 5.1 important text features of the dataset

5.1.5 COMPLEX FEATURES OF THE DATASET

LOCATION

```
location = data['location'].copy()
location.head(15)
location_splitted = list(location.str.split(', ').values)
location_splitted[:15]
for loc_ind, loc in enumerate(location_splitted):
    if loc is np.nan:
        location_splitted[loc_ind] = ['Unspecified'] * 3
```

```
    else:
        for el_ind, el in enumerate(loc):
            if el == '':
                loc[el_ind] = 'Unspecified'
location_splitted[:15]
```

```
···  [['US', 'NY', 'New York'],
      ['NZ', 'Unspecified', 'Auckland'],
      ['US', 'IA', 'Wever'],
      ['US', 'DC', 'Washington'],
      ['US', 'FL', 'Fort Worth'],
      ['US', 'MD', 'Unspecified'],
      ['DE', 'BE', 'Berlin'],
      ['US', 'CA', 'San Francisco'],
      ['US', 'FL', 'Pensacola'],
      ['US', 'AZ', 'Phoenix'],
      ['US', 'NJ', 'Jersey City'],
      ['GB', 'LND', 'London'],
      ['US', 'CT', 'Stamford'],
      ['US', 'FL', 'Orlando'],
      ['AU', 'NSW', 'Sydney']]
```

Fig. 5.2 Locations mentioned in the dataset

LOCATION WITH GREATER THAN OR LESSER PARTS

```
for loc_ind, loc in enumerate(location_splitted):
    if len(loc) > 3:
        print(loc_ind, loc)
for loc_ind, loc in enumerate(location_splitted):
    if len(loc) < 3:
        print(loc_ind, loc)
location_splitted = list(map(lambda loc: list(loc), location_splitted))
for loc_ind, loc in enumerate(location_splitted):
    if len(loc) > 3:
        location_splitted[loc_ind] = loc[:2] + [', '.join(loc[2:])]
    if len(loc) < 3:
        location_splitted[loc_ind] += ['Unpecified'] * 2

data_location = pd.DataFrame(location_splitted, columns=['country', 'state',
'city'])

cat_features += ['country', 'state', 'city']

data = pd.concat([data, data_location], axis=1)
```

```
data.drop('location', axis=1, inplace=True)
data_location.head(15)
```

	country	state	city
0	US	NY	New York
1	NZ	Unspecified	Auckland
2	US	IA	Wever
3	US	DC	Washington
4	US	FL	Fort Worth
5	US	MD	Unspecified
6	DE	BE	Berlin
7	US	CA	San Francisco
8	US	FL	Pensacola
9	US	AZ	Phoenix
10	US	NJ	Jersey City
11	GB	LND	London
12	US	CT	Stamford
13	US	FL	Orlando
14	AU	NSW	Sydney

Fig. 5.3 Locations with greater than or lesser than parts

SALARY RANGE

```
salary_range = data.salary_range.copy()
salary_range.head(15)
salary_range.fillna('0-0', inplace=True)
salary_range_sep = list(salary_range.str.split('-').values)
salary_range_sep[:5]
for range_ind, s_range in enumerate(salary_range_sep):
    if len(s_range) < 2 or len(s_range) > 2:
        print(range_ind, s_range)
salary_range_sep[5538] = ['40000', '40000']
error_range_inds = []
for range_ind, s_range in enumerate(salary_range_sep):
    min_value, max_value = s_range
    if not min_value.isdigit() or not max_value.isdigit():
        print(range_ind, (min_value, max_value))
```

```
        error_range_inds += [range_ind]
for range_ind in error_range_inds:
    salary_range_sep[range_ind] = ['0', '0']
data_salary_range = pd.DataFrame(np.array(salary_range_sep, dtype='int64'),
                                 columns=['min_salary', 'max_salary'])
data_salary_range.head(15)
data_salary_range['salary_specified'] = ((data_salary_range.min_salary != 0) |
                                         (data_salary_range.max_salary !=
0)).astype('int64')
data_salary_range.head(15)
num_features = ['min_salary', 'max_salary']
bin_features += ['salary_specified']
data = pd.concat([data, data_salary_range], axis=1)
data.head()
data.drop('salary_range', axis=1, inplace=True)
data.info()

data.fillna('Unspecified', inplace=True)
data.info()
```

```
...
 23  max_salary                17880 non-null  int64
 24  salary_specified          17880 non-null  int64
dtypes: int32(4), int64(7), object(14)
```

Fig. 5.4 Output where features are not specified

```
f = []
for ind, val in enumerate(data['fraudulent']):
    if val == 1:
        f.append('FRAUD')
    else:
        f.append('REAL')

data.pop('fraudulent')

print(bin_features)
print(text_features)
print(complex_features)
print(cat_features)
print(num_features)
data['fraudulent'] = f
```

culture anything corporatewe collaborative cre...	0	1	0	Full-time	...	1
full benefits offered	0	1	1	Full-time	...	1

Fig. 5.5 Salary Ranges offered by companies in the dataset

5.1.6 SAVING THE CLEANED DATASET TO A NEW CSV FILE

```
data.to_csv('./cleaned-data.csv')
```

5.2 EXPLORATORY DATA ANALYSIS

```
fig = plt.figure(figsize=(25, 30))
outer = gridspec.GridSpec(4, 2, wspace=0.2, hspace=0.1)

for feature_ind, feature_name in enumerate(bin_features):
    inner = gridspec.GridSpecFromSubplotSpec(1, 2, subplot_spec=outer[feature_ind],
                                             wspace=0.5, hspace=0.7)

    ax = plt.Subplot(fig, outer[feature_ind])
    ax.set_title(f'The distribution of fraudulent for each {feature_name}\'s class')
    ax.axis('off')
    fig.add_subplot(ax)

    for feature_class in [0, 1]:
        ax = plt.Subplot(fig, inner[feature_class])
        feature_cl_vc = data[data[feature_name] ==
feature_class].fraudulent.value_counts().sort_index()
        if len(feature_cl_vc) == 2:
            feature_cl_vc.index = ['non-fraudulent', 'fraudulent']
        else:
            feature_cl_vc.index = ['fraudulent']

        ax.pie(feature_cl_vc.values, labels=feature_cl_vc.index, autopct='%1.1f%%')
        ax.set_title(f'{feature_name} = {feature_class}')
        fig.add_subplot(ax)

fig.suptitle('Distributions of fraudulent for the binary features')
fig.subplots_adjust(top=0.95)
fig.show()
```

5.3 CREATING A CONTINGENCY TABLE

```
cont_table = pd.crosstab(data.fraudulent, data.description_specified)
print('Contingency table (fraudulent x description_specified):')
display(cont_table)
```

Contingency table (fraudulent x description_specified):

description_specified	0	1
fraudulent		
0	0	17014
1	1	865

Fig. 5.6 Contingency Table for fraudulent x description feature of the dataset

```
def print_stats_for_texts(feature_name):
    '''Calculates statistics for fraudulent and non-fraudulent count of words in
feature\'s texts.'''
    if feature_name == 'title':
        feature_values_0f = data[(data.fraudulent == 0)][feature_name].astype(str)
        feature_values_1f = data[(data.fraudulent == 1)][feature_name].astype(str)
    else:
        feature_values_0f = data[(data.fraudulent == 0) &
data[f'{feature_name}_specified']][feature_name].astype(str)
        feature_values_1f = data[(data.fraudulent == 1) &
data[f'{feature_name}_specified']][feature_name].astype(str)

    lens_0f = feature_values_0f.str.split(' ').apply(len)
    lens_1f = feature_values_1f.str.split(' ').apply(len)
    mean_lens_0f = round(np.mean(lens_0f), 4)
    mean_lens_1f = round(np.mean(lens_1f), 4)

    bigger_mean, smaller_mean = (lens_0f, lens_1f) if mean_lens_0f > mean_lens_1f
else (lens_1f, lens_0f)
    mean_diff = round(np.mean(bigger_mean) - np.mean(smaller_mean), 4)

    print(f'Feature: {feature_name}\n======')
    print(f'Mean of {feature_name}\'s count of words in non-fraudulent posts:
{mean_lens_0f}')
    print(f'Mean of {feature_name}\'s count of words in fraudulent
posts:     {mean_lens_1f}')
    print(f'Difference in these means: {mean_diff}')
for feature_name in text_features:
    print_stats_for_texts(feature_name)
    print()
```

```
Feature: title
======
Mean of title's count of words in non-fraudulent posts: 3.2794
Mean of title's count of words in fraudulent posts:     3.5612
Difference in these means: 0.2818

Feature: company_profile
======
Mean of company_profile's count of words in non-fraudulent posts: 66.9311
Mean of company_profile's count of words in fraudulent posts:     59.2294
Difference in these means: 7.7017

Feature: description
======
Mean of description's count of words in non-fraudulent posts: 103.8457
Mean of description's count of words in fraudulent posts:     100.0358
Difference in these means: 3.8098

Feature: requirements
======
Mean of requirements's count of words in non-fraudulent posts: 60.1132
Mean of requirements's count of words in fraudulent posts:     47.3244
Difference in these means: 12.7887

Feature: benefits
======
Mean of benefits's count of words in non-fraudulent posts: 30.6589
Mean of benefits's count of words in fraudulent posts:     30.5359
Difference in these means: 0.1231
```

Fig. 5.7 Word Count for the main features of the fraudulent and real postings

```
data['company_profile_count_of_words'] =
data['company_profile'].astype(str).str.split(' ').apply(len)
data['requirements_count_of_words'] = data['requirements'].astype(str).str.split('
').apply(len)
data.head()[['company_profile_count_of_words', 'requirements_count_of_words']]
```

	company_profile_count_of_words	requirements_count_of_words
0	86	73
1	96	117
2	75	103
3	55	121
4	146	61

Fig. 5.8 Word Count for company profile and requirements

5.4 CORRELATION

```
num_features += ['company_profile_count_of_words', 'requirements_count_of_words']
```

```
plt.figure(figsize=(20,20))
plt.show(sns.heatmap(data.corr(), annot=True))
```

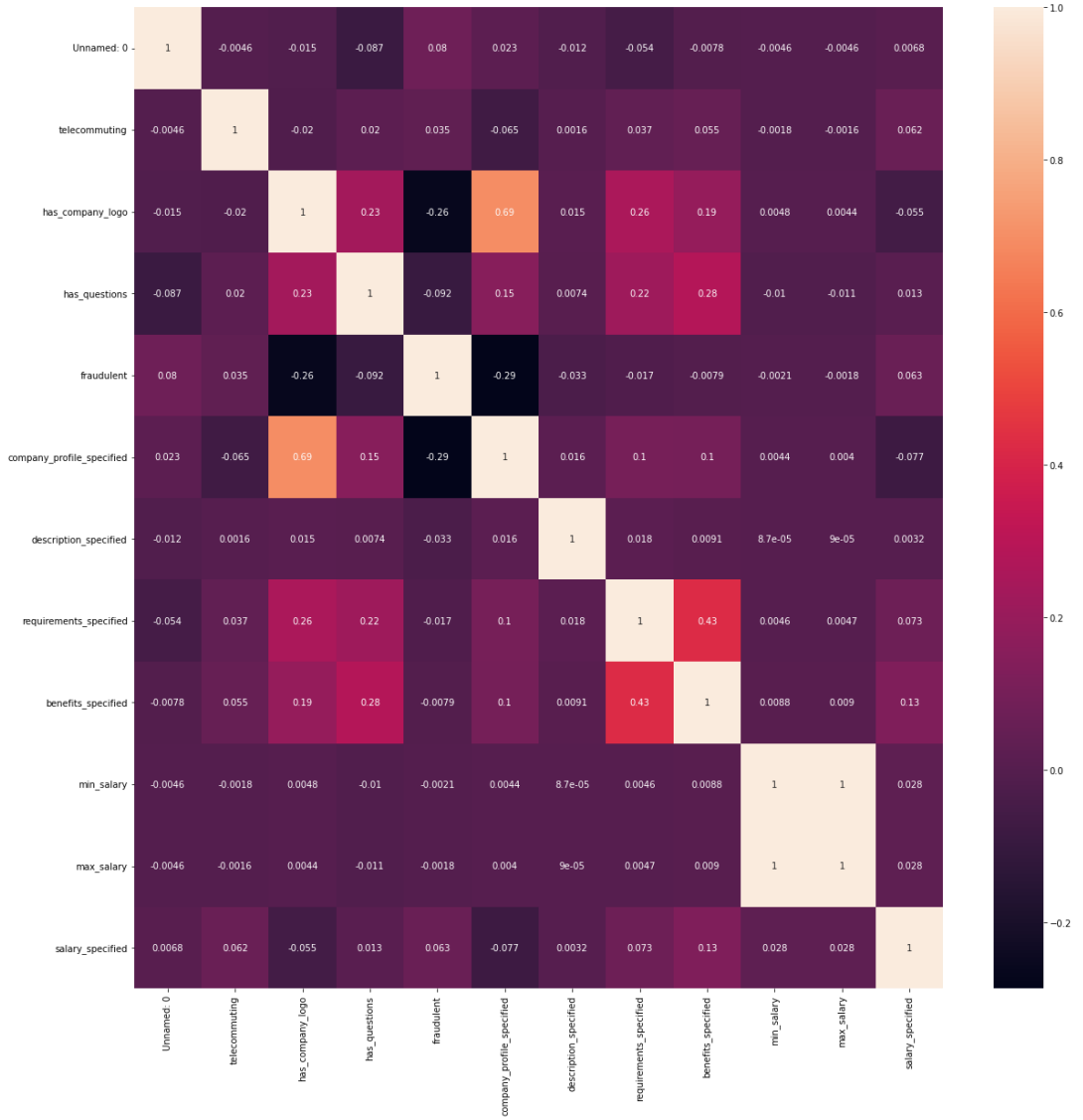

Fig. 5.9 Correlation matrix of all the features

5.5 COMPARISON

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.svm import LinearSVC
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix,
ConfusionMatrixDisplay
from xgboost import XGBClassifier
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
import re
import string
import nltk
from nltk.corpus import stopwords
import matplotlib.pyplot as plt
import seaborn as sns


dataSet = pd.read_csv("../cleaned-data.csv")

Y = np.array(dataSet['fraudulent'])
rf = RandomForestClassifier()
lr = LogisticRegression(max_iter=1000)
svm = LinearSVC(C=0.01, class_weight="balanced", random_state=42)
svm = Pipeline([('svm', svm)])
xgbc = XGBClassifier(objective="binary:logistic")
Y = np.array(dataSet['fraudulent'])



plt.figure(figsize=(20,20))
plt.show(sns.heatmap(dataSet.corr(), annot=True))
```

5.5.1 DATA FEATURE 1 – COMPANY DESCRIPTION

```
X = np.array(dataSet['company_profile'])
newX = ['Unspecified' if x is np.NaN else x for x in X]
X = newX
cv = CountVectorizer()
X = cv.fit_transform(X)
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.33,
random_state=42)


rf.fit(x_train, y_train)
lr.fit(x_train, y_train)
svm.fit(x_train, y_train)
```

```
xgbc.fit(x_train, y_train)

rfP = rf.predict(x_test)
lrP = lr.predict(x_test)
svmP = svm.predict(x_test)
xgbP = xgbc.predict(x_test)
```

RANDOM FOREST CLASSIFIER

```
print("Accuracy :", accuracy_score(y_test, rfP), end="\n\n")
print(classification_report(y_test, rfP))
print("Confusion Matrix")
cm = confusion_matrix(y_test, rfP)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

LOGISTIC REGRESSION

```
print("Accuracy :", accuracy_score(y_test, lrP), end="\n\n")
print(classification_report(y_test, lrP))
print("Confusion Matrix")
cm = confusion_matrix(y_test, lrP)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

SUPPORT VECTOR MACHINE

```
print("Accuracy :", accuracy_score(y_test, svmP), end="\n\n")
print(classification_report(y_test, svmP))
print("Confusion Matrix")
cm = confusion_matrix(y_test, svmP)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

XGBOOST

```
print("Accuracy :", accuracy_score(y_test, xgbP), end="\n\n")
print(classification_report(y_test, xgbP))
print("Confusion Matrix")
cm = confusion_matrix(y_test, xgbP)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

5.5.2 DATA FEATURE 2 – JOB DESCRIPTION

```
X = np.array(dataSet['description'])
newX = ['Unspecified' if x is np.NaN else x for x in X]
X = newX
cv = CountVectorizer()
X = cv.fit_transform(X)
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.33,
random_state=42)
rf.fit(x_train, y_train)
lr.fit(x_train, y_train)
svm.fit(x_train, y_train)
xgbc.fit(x_train, y_train)
rfP = rf.predict(x_test)
lrP = lr.predict(x_test)
svmP = svm.predict(x_test)
xgbP = xgbc.predict(x_test)
```

RANDOM FOREST CLASSIFIER

```
print("Accuracy :", accuracy_score(y_test, rfP), end="\n\n")
print(classification_report(y_test, rfP))
print("Confusion Matrix")
cm = confusion_matrix(y_test, rfP)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

LOGISTIC REGRESSION

```
print("Accuracy :", accuracy_score(y_test, lrP), end="\n\n")
print(classification_report(y_test, lrP))
print("Confusion Matrix")
cm = confusion_matrix(y_test, lrP)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

SUPPORT VECTOR MACHINE

```
print("Accuracy :", accuracy_score(y_test, svmP), end="\n\n")
print(classification_report(y_test, svmP))
print("Confusion Matrix")
cm = confusion_matrix(y_test, svmP)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
```

```
plt.show()
```

XGBOOST

```
print("Accuracy :", accuracy_score(y_test, xgbP), end="\n\n")
print(classification_report(y_test, xgbP))
print("Confusion Matrix")
cm = confusion_matrix(y_test, xgbP)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

5.5.3 DATA FEATURE 3 – REQUIREMENTS

```
X = np.array(dataSet['requirements'])
newX = ['Unspecified' if x is np.NaN else x for x in X]
X = newX
cv = CountVectorizer()
X = cv.fit_transform(X)
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.33,
random_state=42)

rf.fit(x_train, y_train)
lr.fit(x_train, y_train)
svm.fit(x_train, y_train)
xgbc.fit(x_train, y_train)

rfP = rf.predict(x_test)
lrP = lr.predict(x_test)
svmP = svm.predict(x_test)
xgbP = xgbc.predict(x_test)
```

RANDOM FOREST CLASSIFIER

```
print("Accuracy :", accuracy_score(y_test, rfP), end="\n\n")
print(classification_report(y_test, rfP))
print("Confusion Matrix")
cm = confusion_matrix(y_test, rfP)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

LOGISTIC REGRESSION

```
print("Accuracy :", accuracy_score(y_test, lrP), end="\n\n")
print(classification_report(y_test, lrP))
```

```
print("Confusion Matrix")
cm = confusion_matrix(y_test, lrP)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

SUPPORT VECTOR MACHINE

```
print("Accuracy :", accuracy_score(y_test, svmP), end="\n\n")
print(classification_report(y_test, svmP))
print("Confusion Matrix")
cm = confusion_matrix(y_test, svmP)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

XGBOOST

```
print("Accuracy :", accuracy_score(y_test, xgbP), end="\n\n")
print(classification_report(y_test, xgbP))
print("Confusion Matrix")
cm = confusion_matrix(y_test, xgbP)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

5.5.4 DATA FEATURE 4 – BENEFITS

```
X = np.array(dataSet['benefits'])
newX = ['Unspecified' if x is np.NaN else x for x in X]
X = newX
cv = CountVectorizer()
X = cv.fit_transform(X)
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.33,
random_state=42)
rf.fit(x_train, y_train)
lr.fit(x_train, y_train)
svm.fit(x_train, y_train)
xgbc.fit(x_train, y_train)
rfP = rf.predict(x_test)
lrP = lr.predict(x_test)
svmP = svm.predict(x_test)
xgbP = xgbc.predict(x_test)
```

RANDOM FOREST CLASSIFIER

```
print("Accuracy :", accuracy_score(y_test, rfP), end="\n\n")
print(classification_report(y_test, rfP))
print("Confusion Matrix")
cm = confusion_matrix(y_test, rfP)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

LOGISTIC REGRESSION

```
print("Accuracy :", accuracy_score(y_test, lrP), end="\n\n")
print(classification_report(y_test, lrP))
print("Confusion Matrix")
cm = confusion_matrix(y_test, lrP)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

SUPPORT VECTOR MACHINE

```
print("Accuracy :", accuracy_score(y_test, svmP), end="\n\n")
print(classification_report(y_test, svmP))
print("Confusion Matrix")
cm = confusion_matrix(y_test, svmP)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

XGBOOST

```
print("Accuracy :", accuracy_score(y_test, xgbP), end="\n\n")
print(classification_report(y_test, xgbP))
print("Confusion Matrix")
cm = confusion_matrix(y_test, xgbP)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

5.5.5 DATA FEATURE 5 – COMPANY LOGO

```
X = np.array(dataSet['has_company_logo'])
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.33,
random_state=42)
x_train = x_train.reshape(-1,1)
```

```
x_test = x_test.reshape(-1,1)
rf.fit(x_train, y_train)
lr.fit(x_train, y_train)
svm.fit(x_train, y_train)
xgbc.fit(x_train, y_train)
rfP = rf.predict(x_test)
lrP = lr.predict(x_test)
svmP = svm.predict(x_test)
xgbP = xgbc.predict(x_test)
```

RANDOM FOREST CLASSIFIER

```
print("Accuracy :", accuracy_score(y_test, rfP), end="\n\n")
print(classification_report(y_test, rfP))
print("Confusion Matrix")
cm = confusion_matrix(y_test, rfP)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

LOGISTIC REGRESSION

```
print("Accuracy :", accuracy_score(y_test, lrP), end="\n\n")
print(classification_report(y_test, lrP))
print("Confusion Matrix")
cm = confusion_matrix(y_test, lrP)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

SUPPORT VECTOR MACHINE

```
print("Accuracy :", accuracy_score(y_test, svmP), end="\n\n")
print(classification_report(y_test, svmP))
print("Confusion Matrix")
cm = confusion_matrix(y_test, svmP)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

XGBOOST

```
print("Accuracy :", accuracy_score(y_test, xgbP), end="\n\n")
print(classification_report(y_test, xgbP))
```

```
print("Confusion Matrix")
cm = confusion_matrix(y_test, xgbP)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

5.5.6 DATA FEATURE 6 – EMPLOYMENT TYPE

```
X = np.array(dataSet['city'])
newX = ['Unspecified' if x is np.NaN else x for x in X]
X = newX
cv = CountVectorizer()
X = cv.fit_transform(X)
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.33,
random_state=42)
rf.fit(x_train, y_train)
lr.fit(x_train, y_train)
svm.fit(x_train, y_train)
xgbc.fit(x_train, y_train)
rfP = rf.predict(x_test)
lrP = lr.predict(x_test)
svmP = svm.predict(x_test)
xgbP = xgbc.predict(x_test)
```

RANDOM FOREST CLASSIFIER

```
print("Accuracy :", accuracy_score(y_test, rfP), end="\n\n")
print(classification_report(y_test, rfP))
print("Confusion Matrix")
cm = confusion_matrix(y_test, rfP)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

LOGISTIC REGRESSION

```
print("Accuracy :", accuracy_score(y_test, lrP), end="\n\n")
print(classification_report(y_test, lrP))
print("Confusion Matrix")
cm = confusion_matrix(y_test, lrP)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

SUPPORT VECTOR MACHINE

```
print("Accuracy :", accuracy_score(y_test, svmP), end="\n\n")
print(classification_report(y_test, svmP))
print("Confusion Matrix")
cm = confusion_matrix(y_test, svmP)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

XGBOOST

```
print("Accuracy :", accuracy_score(y_test, xgbP), end="\n\n")
print(classification_report(y_test, xgbP))
print("Confusion Matrix")
cm = confusion_matrix(y_test, xgbP)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

5.6 CREATING THE ENSEMBLE MODEL

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.pipeline import Pipeline
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
ConfusionMatrixDisplay, classification_report
from sklearn.model_selection import train_test_split
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer
import pandas as pd
import numpy as np
import joblib
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
import warnings
warnings.filterwarnings('ignore')
nltk_lang = stopwords.fileids()
stop_words = set(stopwords.words(nltk_lang))
weightRF = 1
weightLR = 1
weightSV = 1
weightXG = 1
threshHold = 2
class CustomVotingClassifier:
    def __init__(self) -> None:
```

```
        self.rf = RandomForestClassifier()
        self.lr = LogisticRegression(max_iter=1000)
        self.svm = LinearSVC(C=0.01, random_state=42)
        self.svm = Pipeline([("SVM", self.svm)])
        self.xgbc = XGBClassifier()
        self.porter = PorterStemmer()

    def preprocesstexts(self, texts):
        preprocess_texts = []
        for text in texts:
            text = ''.join([sym.lower() for sym in text if sym.isalpha() or sym == '
']) 
            tokenized_text = word_tokenize(text)
            tokenized_text_wo_sw = [word for word in tokenized_text if word not in
stop_words]
            tokenized_text_wo_sw_stem = [self.porter.stem(word) for word in
tokenized_text_wo_sw]
            preprocess_texts += [' '.join(tokenized_text_wo_sw_stem)]
        return preprocess_texts


    def fit(self,  xtrain1, xtrain2, xtrain3, xtrain4, ytrain) :
        self.rf.fit(xtrain1, ytrain)
        self.lr.fit(xtrain2, ytrain)
        self.svm.fit(xtrain3, ytrain)
        self.xgbc.fit(xtrain4, ytrain)

    def predict(self, data):
        # self.data = data.apply(self.clean)
        result = []
        self.rfP = np.array(self.rf.predict(data))
        self.lrP = np.array(self.lr.predict(data))
        self.svmP = np.array(self.svm.predict(data))
        self.xgbP = np.array(self.xgbc.predict(data))

        for i in range(len(self.rfP)):
            if (self.rfP[i] * weightRF + self.lrP[i] * weightLR + self.svmP[i] *
weightSV + self.xgbP[i] * weightXG) >= threshHold:
                result.append(1)
            else:
                result.append(0)

        return result

    def classification(self, y_test):
        print("RF")
        print(classification_report(y_test, self.rfP))
```

```
        print(accuracy_score(y_test, self.rfP))
        print("LR")
        print(classification_report(y_test, self.lrP))
        print(accuracy_score(y_test, self.lrP))
        print("SVM")
        print(classification_report(y_test, self.svmP))
        print(accuracy_score(y_test, self.svmP))
        print("XGB")
        print(classification_report(y_test, self.xgbP))
        print(accuracy_score(y_test, self.xgbP))

CVC = CustomVotingClassifier()
cv = CountVectorizer()


dataSet = pd.read_csv("../Datasets/cleaned-data.csv")

cp = dataSet['company_profile']
d = dataSet['description']
b = dataSet['benefits']
r = dataSet['requirements']
newCP = ['Unspecified' if x is np.NaN else x for x in cp]
newD = ['Unspecified' if x is np.NaN else x for x in d]
newB = ['Unspecified' if x is np.NaN else x for x in b]
newR = ['Unspecified' if x is np.NaN else x for x in r]
X = []
for i in range(len(newCP)):
    X.append(newCP[i] + " " + newD[i]+ " " + newB[i] + " " + newR[i])
X = np.array(X)
Y = dataSet['fraudulent']

cv = CountVectorizer()
X = cv.fit_transform(X)
xtrain, xtest, ytrain, ytest = train_test_split(X, Y, test_size=0.3,
random_state=42)
print("Training")
CVC.fit(xtrain1=xtrain, xtrain2=xtrain, xtrain3=xtrain, xtrain4=xtrain,
ytrain=ytrain)
print("result")
# inputD = input()
# inputD = CVC.clean(inputD)
# CVC.predict(cv.transform([inputD]).toarray())
CVC.predict(xtest)
print(classification_report(ytest, CVC.predict(xtest)))
```

Training

	precision	recall	f1-score	support
0	0.99	1.00	0.99	5093
1	0.97	0.72	0.82	271
accuracy			0.98	5364
macro avg	0.98	0.86	0.91	5364
weighted avg	0.98	0.98	0.98	5364

Fig. 5.10 Training the ensemble model

```
print(classification_report(ytest, CVC.predict(xtest)))
print(accuracy_score(ytest, CVC.predict(xtest)))
CVC.classification(ytest)
k = CVC.preprocesstexts([input()])
newK = ['Unspecified' if x is np.NaN else x for x in k]
print(newK)
asd = cv.transform(newK).toarray()
print(CVC.predict(asd))
```

All the files we have worked with so far and the completed model are then made into pkl files and deployed into flask for testing and tuning to make adjustments and see if the model is working as intended. A pickle is largely used in Python for serializing and deserializing Python object structures. In other words, it is the act of transforming an object into something like a byte flow in order to store it in a file/database, retain program state between sessions, or transmit information over a network.

5.7 TUNING THE MODEL

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.pipeline import Pipeline
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from nltk.corpus import stopwords
```

```
from sklearn.feature_extraction.text import CountVectorizer
import pandas as pd
import numpy as np
import joblib
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
import warnings
import nltk
nltk.download("stopwords")
warnings.filterwarnings('ignore')
nltk_lang = stopwords.fileids()
stop_words = set(stopwords.words(nltk_lang))


rf = RandomForestClassifier()
lr = LogisticRegression(max_iter=1000)
svm = LinearSVC(C=0.01, random_state=42)
svm = Pipeline([("SVM", svm)])
xgbc = XGBClassifier()
porter = PorterStemmer()


dataSet = pd.read_csv("cleaned-data.csv")

cp = dataSet['company_profile']
d = dataSet['description']
b = dataSet['benefits']
r = dataSet['requirements']
newCP = ['Unspecified' if x is np.NaN else x for x in cp]
newD = ['Unspecified' if x is np.NaN else x for x in d]
newB = ['Unspecified' if x is np.NaN else x for x in b]
newR = ['Unspecified' if x is np.NaN else x for x in r]
X = []
for i in range(len(newCP)):
    X.append(newCP[i] + " " + newD[i]+ " " + newB[i] + " " + newR[i])
gg = X
X = pd.Series(X)
Y = dataSet['fraudulent']

cv = CountVectorizer()
X = cv.fit_transform(X)
xtrain, xtest, ytrain, ytest = train_test_split(X, Y, test_size=0.3,
random_state=42)


rf.fit(xtrain, ytrain)
lr.fit(xtrain, ytrain)
```

```
svm.fit(xtrain, ytrain)
xgbc.fit(xtrain, ytrain)


rfP = rf.predict(xtest)
lrP = lr.predict(xtest)
svmP = svm.predict(xtest)
xgbP = xgbc.predict(xtest)


from sklearn.metrics import f1_score
from tqdm import tqdm
def tune():
    mA = 0
    mF = 0
    maxAcc = []
    maxF1 = []
    for i in tqdm(range(0,11,1)):
        for j in (range(0,11,1)):
            for k in (range(0,11,1)):
                for l in (range(0,11,1)):
                    for m in (range(1,40,1)):
                        res = []
                        for g in range(len(rfP)):
                            if (rfP[g]*i+ lrP[g]*j + svmP[g]*k + xgbP[g]*l) >= m:

                                res.append(1)
                            else:
                                res.append(0)
                        acc = accuracy_score(ytest, res)
                        f = f1_score(ytest, res)
                        if acc >= mA:
                            maxAcc.append([i,j,k,l,m,acc])
                            mA = acc
                        if f >= mF:
                            maxF1.append([i,j,k,l,m,f])
                            mF = f
    return mA, mF, maxAcc, maxF1




maxAcc[-1]
maxF1[-1]
```

```
[10, 9, 10, 10, 10, 0.9865771812080537]

[10, 10, 10, 10, 10, 0.8542914171656687]
```

Fig. 5.11 Accuracy and F1 Scores of the tuned model

```
import csv
heading = ['RF','LR','SV','XG','MAX']
with open("maxACC.csv", 'w') as f:
    write = csv.writer(f)
    write.writerow(heading)
    write.writerows(maxAcc)
with open("maxF1.csv", 'w') as f:
    write = csv.writer(f)
    write.writerow(heading)
    write.writerows(maxF1)
```

CHAPTER 6

RESULTS AND OBSERVATIONS

6.1. COMBINING ALL FEATURES:

Accuracy and f1-score of all models with combined data as follows.

6.1.1. RANDOME FOREST CLASSIFIER

	precision	recall	f1-score	support
0	0.95	1.00	0.98	5113
1	0.00	0.00	0.00	251
accuracy			0.95	5364
macro avg	0.48	0.50	0.49	5364
weighted avg	0.91	0.95	0.93	5364

Fig 6.1 Performance of RFC with Combined Features

6.1.2. LOGISTIC REGRESSION

	precision	recall	f1-score	support
0	0.95	1.00	0.98	5113
1	0.00	0.00	0.00	251
accuracy			0.95	5364
macro avg	0.48	0.50	0.49	5364
weighted avg	0.91	0.95	0.93	5364

Fig 6.2 Performance of LR with Combined Features

6.1.3. SVM

	precision	recall	f1-score	support
0	0.98	0.77	0.86	5113
1	0.12	0.65	0.20	251
accuracy			0.76	5364
macro avg	0.55	0.71	0.53	5364
weighted avg	0.94	0.76	0.83	5364

Fig 6.3 Performance of SVM with Combined Features

6.1.4. XGBOOST

	precision	recall	f1-score	support
0	0.95	1.00	0.98	5113
1	0.00	0.00	0.00	251
accuracy			0.95	5364
macro avg	0.48	0.50	0.49	5364
weighted avg	0.91	0.95	0.93	5364

Fig 6.4 Performance of XGBoost with Combined Features

6.2. PERFORMANCE OF ENSEMBLE MODEL

Now that the ensemble model is created, we are measuring the performance of the ensemble model.

6.2.1. BEFORE TUNING

In Table 6.1, the performance of our ensemble model is listed. These results are obtained before tuning.

Table 6.1 Performance of Ensemble model before Tuning the model

Performance Metrics	**Models**
	Custom Ensemble Model
Precision	0.97
Recall	0.72
F1 - Score	0.82
Accuracy	98.4%

6.2.2. AFTER TUNING

In Table 6.2, the performance of our ensemble model is listed. These results are obtained after tuning the model.

Table 6.2 Performance of Ensemble model after Tuning the model

Performance Metrics	**Models**
	Custom Ensemble Model
recision	0.93
Recall	0.79
F1 - Score	0.86
Accuracy	98.6%

6.2.3. OVERALL PERFORMANCE

In fig the overall performance of the ensemble model is listed.

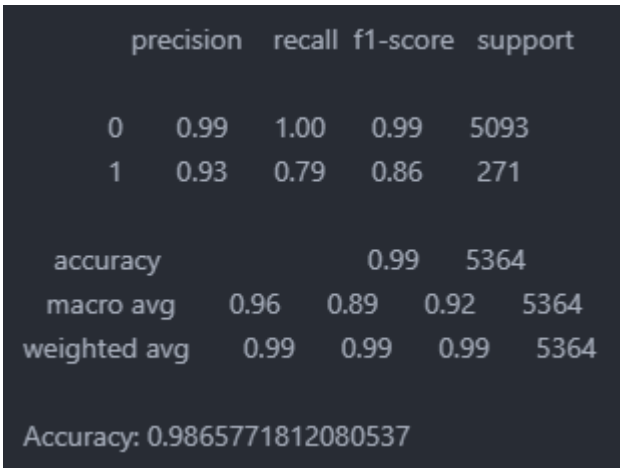

Fig 6.5 Performance of Ensemble Model with Combined Features

0 – real job posting

1 – fake job posting

The confusion matrix is drawn for real and job posting in fig.

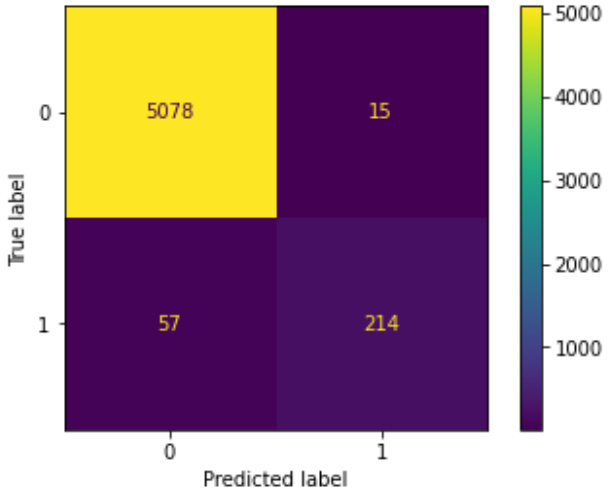

Fig 6.6 Confusion matrix of Ensemble Model with Combined Features

6.3. OUTPUT SCREENSHOTS

Now we can look at fig 6.7, 6.8, 6.9 to view how the final product looks like.

6.3.1 INPUT SCREEN

This is the input screen where the user has to enter the inputs of job posting

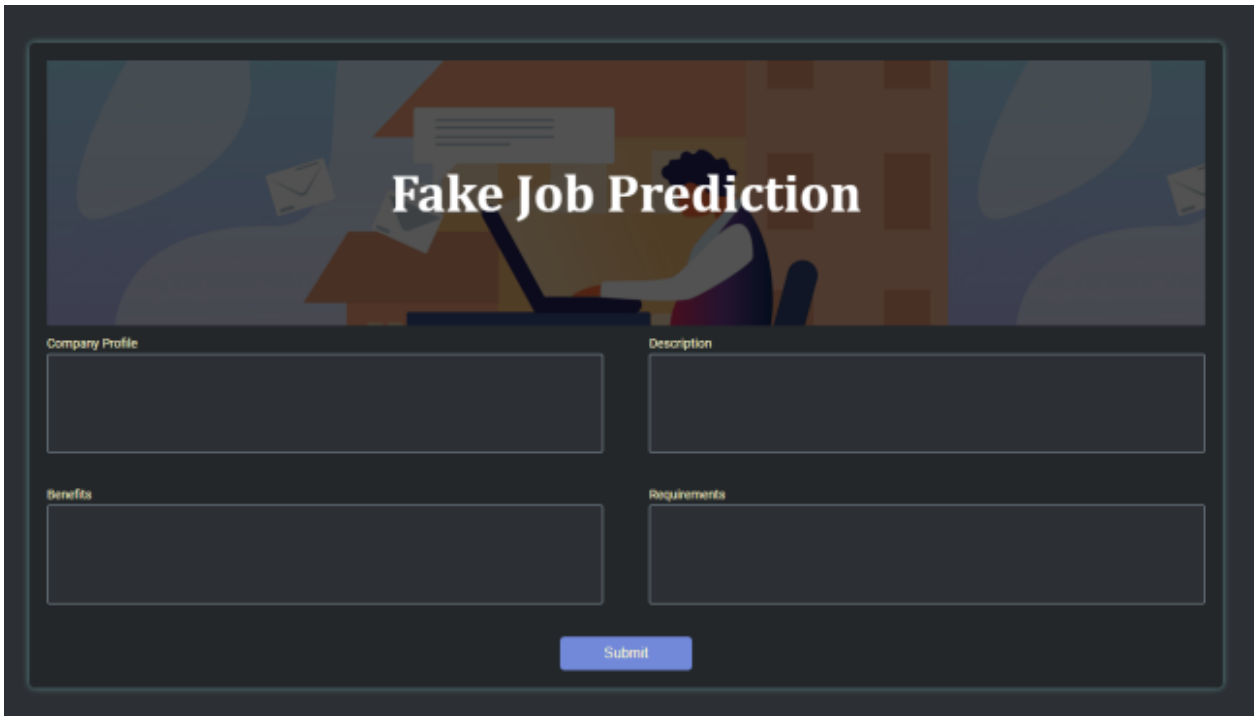

Fig 6.7 Input Screen.

6.3.2 OUTPUT SCREENS

This screen is displayed when the job posting that the user entered is fake.

Fig 6.8 Fake Job Output Screen

The following page is shown when the job posting is real.

Fig 6.9 Real Job Output Screen

CONCLUSION

In this project we have created a fraudulent checker tool which uses an ensemble model – combination of 4 Machine Learning Algorithms - SVM, XGBoost, Logistic Regression, and Random Forest Classifier. This ensemble model shows an accuracy of 98.6% and F1 score of 0.99 and 0.85 for non-fraudulent and fraudulent respectively. We first did data preprocessing where the quality of raw data is checked, cleaned, transformed, and reduced into an understandable format. It has 4 major steps: data cleaning, integration, reduction, and transformation. Then exploratory Data Analysis takes place. Exploratory Data Analysis is a core step for discovering patterns and anomalies in the dataset and form hypotheses based on our understanding of it. Running the input data through the algorithm to correlate the processed output with the sample output is how the model is trained. The model is modified based on the results of this association. The four models are compared.

4 ML classification models - Individually, the Random Forest Classifier, Logistic Regression, XGBoost, and Support Vector Machines are assessed against four separate aspects of the EMSCAD dataset (Company profile, job description, benefits, and requirements). The cleaned dataset is split into separate training and testing datasets in such a way that the number of real and fraudulent jobs are balanced. The 5 models are then trained to differentiate and identify the nature of the jobs. The cleaned dataset is split into separate training and testing datasets in such a way that the number of real and fraudulent jobs are balanced. The 5 models are then trained to differentiate and identify the nature of the jobs. Finally, the Ensemble Model thus created is deployed and connected to a Flask application which is the interface the user will interact with and enter details which the model will analyze such as the company’s profile, the job description, benefits, and requirements.

FUTURE WORK

This project can be further enhanced in the future by hosting the website in cloud and make it so that it redirects to the source of the job posting if the job posting is real. Another enhancement that could be done is that this entire application can be connected to cloud and can be added as an extension where, whenever the client looks at a job posting, the application can alert the user if the job posting is fake, or the user can manually run the application to find whether the job posting is fake or not. The application can also list the reasons why the job posting could be fake, and the fake percentage. With this, the user can easily determine why the job posting is fake and know the fake percentage.

REFERENCES

[1] F. Martínez-Plumed et al., "CRISP-DM Twenty Years Later: From Data Mining Processes to Data Science Trajectories," in IEEE Transactions on Knowledge and Data Engineering, vol. 33, no. 8, pp. 3048-3061, 1 Aug. 2021, doi: 10.1109/TKDE.2019.2962680.

[2] E. Cambria and B. White, "Jumping NLP Curves: A Review of Natural Language Processing Research [Review Article]," in IEEE Computational Intelligence Magazine, vol. 9, no. 2, pp. 48-57, May 2014, doi: 10.1109/MCI.2014.2307227.

[3] M. Nazar, M. M. Alam, E. Yafi and M. M. Su'ud, "A Systematic Review of Human–Computer Interaction and Explainable Artificial Intelligence in Healthcare With Artificial Intelligence Techniques," in IEEE Access, vol. 9, pp. 153316-153348, 2021, doi: 10.1109/ACCESS.2021.3127881.

[4] F. O. Olowononi, D. B. Rawat and C. Liu, "Resilient Machine Learning for Networked Cyber Physical Systems: A Survey for Machine Learning Security to Securing Machine Learning for CPS," in IEEE Communications Surveys & Tutorials, vol. 23, no. 1, pp. 524-552, Firstquarter 2021, doi: 10.1109/COMST.2020.3036778.

[5] L. Yu, S. Gan, Y. Chen and M. He, "Correlation-Based Weight Adjusted Naive Bayes," in IEEE Access, vol. 8, pp. 51377-51387, 2020, doi: 10.1109/ACCESS.2020.2973331.

[6] R. Nock, W. B. H. Ali, R. D'Ambrosio, F. Nielsen and M. Barlaud, "Gentle Nearest Neighbors Boosting over Proper Scoring Rules," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 37, no. 1, pp. 80-93, Jan. 2015, doi: 10.1109/TPAMI.2014.2307877.

[7] K. Taunk, S. De, S. Verma and A. Swetapadma, "A Brief Review of Nearest Neighbor Algorithm for Learning and Classification," 2019 International Conference on Intelligent Computing and Control Systems (ICCS), 2019, pp. 1255-1260, doi: 10.1109/ICCS45141.2019.9065747.

[8] S. Jeong and J. Lee, "Modulation Code and Multilayer Perceptron Decoding for Bit-Patterned Media Recording," in IEEE Magnetics Letters, vol. 11, pp. 1-5, 2020, Art no. 6502705, doi: 10.1109/LMAG.2020.2993000.

[9] L. Abhishek, "Optical Character Recognition using Ensemble of SVM, MLP and Extra Trees Classifier," 2020 International Conference for Emerging Technology (INCET), 2020, pp. 1-4, doi: 10.1109/INCET49848.2020.9154050.

[10] Z. Li, G. Liu, and C. Jiang, "Deep Representation Learning with Full Center Loss for Credit Card Fraud Detection," in IEEE Transactions on Computational Social Systems, vol. 7, no. 2, pp. 569-579, April 2020, doi: 10.1109/TCSS.2020.2970805.

[11] D. Ranparia, S. Kumari and A. Sahani, "Fake Job Prediction using Sequential Network," 2020 IEEE 15th International Conference on Industrial and Information Systems (ICIIS), 2020, pp. 339-343, doi: 10.1109/ICIIS51140.2020.9342738.

[12] Vidros, Sokratis, Constantinos Kolias, Georgios Kambourakis, and Leman Akoglu. 2017. "Automatic Detection of Online Recruitment Frauds: Characteristics, Methods, and a Public Dataset" *Future Internet* 9, no. 1: 6

[13] D. J. Ladani and N. P. Desai, "Stopword Identification and Removal Techniques on TC and IR applications: A Survey," 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), 2020, pp. 466-472, doi: 10.1109/ICACCS48705.2020.9074166.

[14] A. A. Kshirsagar and P. A. Deshkar, "Review analyzer analysis of product reviews on WEKA classifiers," 2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), 2015, pp. 1-5, doi: 10.1109/ICIIECS.2015.7193034.

[15] T. Wisanwanichthan and M. Thammawichai, "A Double-Layered Hybrid Approach for Network Intrusion Detection System Using Combined Naive Bayes and SVM," in IEEE Access, vol. 9, pp. 138432-138450, 2021, doi: 10.1109/ACCESS.2021.3118573.

[16] A. M. Kibriya, E. Frank, B. Pfahringer, and G. Holmes, ―Multinomial naive bayes for text categorization revisited, ‖ in Australasian Joint Conference on Artificial Intelligence, 2004, pp. 488–499.

[17] S. J. RUSSEL and P. Norvig, ―Artificial Intelligence–A Modern Approach. Person Edu ca tion, ‖ Inc., New Jersey, pp. 736– 741, 2003

[18] A. McCallum and K. Nigam, ―A comparison of event models for naive bayes text classification, ‖ in AAAI-98 workshop on learning for text categorization, 1998, vol. 752, no. 1, pp. 41–48

[19] Q. Zhu, X. Ma and X. Li, "Statistical learning for semantic parsing: A survey," in Big Data Mining and Analytics, vol. 2, no. 4, pp. 217-239, Dec. 2019, doi: 10.26599/BDMA.2019.9020011.

[20] C. J. C. Burges, ―A tutorial on support vector machines for pattern recognition, ‖ Data

Min. Knowl. Discov., vol. 2, no. 2, pp. 121–167, 1998.

[21] N. M. Tahir, A. Hussain, S. A. Samad, K. A. Ishak, and R. A. Halim, ―Feature selection for classification using decision tree, ‖ in 2006 4th Student Conference on Research and Development, 2006, pp. 99–102.

[22] C. Eyupoglu, ―Implementation of color face recognition using PCA and k-NN classifier, ‖ in 2016 IEEE NW Russia Young Researchers in Electrical and Electronic Engineering Conference (EIConRusNW), 2016, pp. 199–202.

[23] J. Wang, P. Neskovic, and L. N. Cooper, ―Improving nearest neighbor rule with a simple adaptive distance measure, ‖ Pattern Recognit. Lett., vol. 28, no. 2, pp. 207–213, 2007.

[24] N. Zhang, J. Yang, and J.-J. Qian, ―Component-based global k-NN classifier for small sample size problems, ‖ Pattern Recognit. Lett., vol. 33, no. 13, pp. 1689–1694, 2012.

[25] L. Breiman, ―Random forests, machine learning 45, ‖ J. Clin. Microbiol, vol. 2, no. 30, pp. 199–228, 2001.

[26] Bansal, ― [Real or Fake] Fake Job Posting Prediction | Kaggle. ‖ https://www.kaggle.com/shivamb/real-or-fake-fakejob posting-prediction (accessed Jun. 03, 2020).

[27] L.-P. Jing, H.-K. Huang, and H.-B. Shi, ―Improved feature selection approach TFIDF in text mining, ‖ in Proceedings. International Conference on Machine Learning and Cybernetics, 2002, vol. 2, pp. 944–946.

[28] V. Kalra and R. Aggarwal, ―Importance of Text Data Preprocessing & Implementation in RapidMiner, ‖ in Proceedings of the First International Conference on Information Technology and Knowledge Management–New Delhi, India, 2017, vol. 14, pp. 71–75.

[29] M. P. Akhter, Z. Jiangbin, I. R. Naqvi, M. Abdelmajeed, A. Mehmood and M. T. Sadiq, "Document-Level Text Classification Using Single-Layer Multisize Filters Convolutional Neural Network," in IEEE Access, vol. 8, pp. 42689-42707, 2020, doi: 10.1109/ACCESS.2020.2976744.

[30] Bandyopadhyay, Samir & Dutta, Shawni. (2020). Fake Job Recruitment Detection Using Machine Learning Approach. International Journal of Engineering Trends and Technology. 68. 10.14445/22315381/IJETT-V68I4P209S.

[31] I. Rish, ―An Empirical Study of the Naïve Bayes Classifier An empirical study of the naive Bayes classifier, ‖ no. January 2001, pp. 41–46, 2014.

[32] D. E. Walters, ―Bayes's Theorem and the Analysis of Binomial Random Variables, ‖ Biometrical J., vol. 30, no. 7, pp. 817–825, 1988, doi: 10.1002/bimj.4710300710.

[33] F. Murtagh, ―Multilayer perceptrons for classification and regression, ‖ Neurocomputing, vol. 2, no. 5–6, pp. 183–197, 1991, doi: 10.1016/0925-2312(91)90023-5

[34] P. Cunningham and S. J. Delany, ―K -Nearest Neighbour Classifiers, ‖ Mult. Classif. Syst., no. May, pp. 1–17, 2007, doi: 10.1016/S0031-3203(00)00099-6.

[35] H. Sharma and S. Kumar, ―A Survey on Decision Tree Algorithms of Classification in Data Mining, ‖ Int. J. Sci. Res., vol. 5, no. 4, pp. 2094–2097, 2016, doi: 10.21275/v5i4.nov162954.

[36] B. Biggio, I. Corona, G. Fumera, G. Giacinto, and F. Roli, ―Bagging classifiers for fighting poisoning attacks in adversarial classification tasks," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 6713 LNCS, pp. 350–359, 2011, doi: 10.1007/978-3-642-21557-5_37.

[37] C. Liu, J. Han, Y. Shang, C. Liu, B. Cheng, and J. Chen, "Predicting of Job Failure in Compute Cloud Based on Online Extreme Learning Machine: A Comparative Study," in IEEE Access, vol. 5, pp. 9359-9368, 2017, doi: 10.1109/ACCESS.2017.2706740.

[38] P. Minet, É. Renault, I. Khoufi and S. Boumerdassi, "Data Analysis of a Google Data Center," 2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), 2018, pp. 342-343, doi: 10.1109/CCGRID.2018.00049.

[39] Yang Le, Zhang Run. Online sequential ELM algorithm and its improvement [J]. Journal of Northwest University: Natural Science Edition, 2012, 42(6): 885－896.

[40] Liang N Y, Huang G B, Saratchandran P, et al. A fast and accurate online sequential learning algorithm for feedforward networks [J]. IEEE Transactions on Neural Networks, 2006, 17(6):1411-23

[41] Mao W, Zhao S, Mu X. Multi-dimensional extreme learning machine [J]. Neurocomputing, 2015, 149(PA):160-170.

[42] Mao W, Xu J, Wang C. A fast and robust model selection algorithm for multi-input multi-output support vector machine [J]. Neurocomputing, 2014, 130 (3):10-19.

[43] Ding S, Qi B. An Overview on Theory and Algorithm of Support Vector Machines [J]. Journal of University of Electronic Science and Technology of China, 2011, 40(01):2-10.

[44] Lau K W, Wu Q H. Online training of support vector classifier [J]. Pattern Recognition,

2003, 36(8):1913 -1920.

[45] Sederberg T W. Free-form deformation of solid geometric models [J]. Acm Siggraph Computer Graphics, 1986, 20(4):151-160.

Plag Check D-6 Final.pdf

ORIGINALITY REPORT

2%	2%	0%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to University of North Texas Student Paper	1%
2	Kaushik Dhola, Mann Saradva. "A Comparative Evaluation of Traditional Machine Learning and Deep Learning Classification Techniques for Sentiment Analysis", 2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 2021 Publication	<1%
3	Submitted to University of Hull Student Paper	<1%
4	rdrr.io Internet Source	<1%
5	www.coursehero.com Internet Source	<1%
6	ir.uitm.edu.my Internet Source	<1%
7	www.ncbi.nlm.nih.gov Internet Source	<1%

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY (Deemed to be University u/a 3 of UGC Act, 1956)		
Office Of Controller of Examinations		
REPORT FOR PLAGARISM CHECKER ON THE DISSERTATION/PROJECT REPORTS FOR UG/PG PROGRAMMES (To be attached in the dissertation/project reports)		
1	Name of the Candidate (**IN BLOCK LETTERS)**	
2	Address of the Candidate	Mobile Number:
3	Registration Number	
4	Date of Birth	
5	Department	
6	Faculty	
7	Title of the Dissertation/Project	Detecting Fake Job Posting Using ML Classifications and Ensemble Model
8	Whether the above Dissertation/Project done by	Individual or group (Strike whichever is not applicable)
		a) If the project / dissertation is done in group, then how many students together completed the project:
		b) Mention the name and register number of other candidates: i. ii.
9	Name and address of the Supervisor / Guide	Mail ID: Mobile Number:
10	Name and address of the Co - Supervisor / Co – Guide (if any)	Mail ID: Mobile Number:
11	Software Used	

12	Data of Verification			
13	**Plagiarism Details: (to attach the final report from the software)**			
Chapter	**Title of the chapter**	**Percentage of similarity index (Including self-citation)**	**Percentage of similarity index (Excluding self-citation)**	**% of plagiarism after excluding Quotes, Bibliography, etc.,**
1				
2				
3				
4				
5				
6				
7				
8				
9				
Appendices				

I / We declare that the above information has been verified and found true to the best of my / our knowledge

Signature of the Candidate	**Name & Signature of the Staff** **(Who uses the plagiarism check software)**
Name and Signature of the Supervisor / Guide	**Name and Signature of the Co - Supervisor / Co – Guide (if any)**

Name and Signature of the HOD

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Deemed to be University u/a 3 of UGC Act, 1956)

Office Of Controller of Examinations

REPORT FOR PLAGARISM CHECKER ON THE DISSERTATION/PROJECT REPORTS FOR UG/PG PROGRAMMES

(To be attached in the dissertation/project reports)

1	Name of the Candidate (**IN BLOCK LETTERS)**	
2	Address of the Candidate	Mobile Number:
3	Registration Number	
4	Date of Birth	
5	Department	
6	Faculty	
7	Title of the Dissertation/Project	Detecting Fake Job Posting Using ML Classifications and Ensemble Model
8	Whether the above Dissertation/Project done by	Individual or group (Strike whichever is not applicable)
		a) If the project / dissertation is done in group, then how many students together completed the project:
		b) Mention the name and register number of other candidates: i. ii.
9	Name and address of the Supervisor / Guide	Mail ID: Mobile Number:
10	Name and address of the Co - Supervisor / Co – Guide (if any)	Mail ID: Mobile Number:
11	Software Used	

12	Data of Verification			
13	**Plagiarism Details: (to attach the final report from the software)**			
Chapter	**Title of the chapter**	**Percentage of similarity index (Including self-citation)**	**Percentage of similarity index (Excluding self-citation)**	**% of plagiarism after excluding Quotes, Bibliography, etc.,**
1				
2				
3				
4				
5				
6				
7				
8				
9				
Appendices				

I / We declare that the above information has been verified and found true to the best of my / our knowledge

Signature of the Candidate	**Name & Signature of the Staff** **(Who uses the plagiarism check software)**
Name and Signature of the Supervisor / Guide	**Name and Signature of the Co - Supervisor / Co – Guide (if any)**

Name and Signature of the HOD

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY (Deemed to be University u/a 3 of UGC Act, 1956)		
Office Of Controller of Examinations		
REPORT FOR PLAGARISM CHECKER ON THE DISSERTATION/PROJECT REPORTS FOR UG/PG PROGRAMMES (To be attached in the dissertation/project reports)		
1	Name of the Candidate (**IN BLOCK LETTERS)**	
2	Address of the Candidate	Mobile Number:
3	Registration Number	
4	Date of Birth	
5	Department	
6	Faculty	
7	Title of the Dissertation/Project	Detecting Fake Job Posting Using ML Classifications and Ensemble Model
8	Whether the above Dissertation/Project done by	Individual or group (Strike whichever is not applicable)
		a) If the project / dissertation is done in group, then how many students together completed the project:
		b) Mention the name and register number of other candidates: i. ii.
9	Name and address of the Supervisor / Guide	Mail ID: Mobile Number:
10	Name and address of the Co - Supervisor / Co – Guide (if any)	Mail ID: Mobile Number:

11	Software Used			
12	Data of Verification			
13	**Plagiarism Details: (to attach the final report from the software)**			
Chapter	**Title of the chapter**	**Percentage of similarity index (Including self-citation)**	**Percentage of similarity index (Excluding self-citation)**	**% of plagiarism after excluding Quotes, Bibliography, etc.,**
1				
2				
3				
4				
5				
6				
7				
8				
9				
Appendices				

I / We declare that the above information has been verified and found true to the best of my / our knowledge

Signature of the Candidate	**Name & Signature of the Staff** **(Who uses the plagiarism check software)**
Name and Signature of the Supervisor / Guide	**Name and Signature of the Co - Supervisor / Co – Guide (if any)**

Name and Signature of the HOD

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956
VADAPALANI

International Research Conference on IoT, Cloud and Data Science (IRCICD'22)

INSTITUTION'S INNOVATION COUNCIL
(Ministry of HRD Initiative)

Certificate of Participation

This is to certify that

Dr./Mr./Mrs./Ms. Rachanna Deva Murali

of SRM INSTITUTE OF SCIENCE AND TECHNOLOGY , VADAPALANI , CHENNAI

has participated and presented a **paper titled**

DETECTING FAKE JOB POSTING USING ML CLASSIFICATIONS & ENSEMBLE MODEL

in the Virtual International Research Conference on IoT, Cloud and Data Science (IRCICD'22) Organised by the Department of Computer Science and Engineering, SRM Institute of Science and Technology, Vadapalani, Chennai, Tamil Nadu, India held on 6th and 7th May 2022.

Dr. Golda Dilip	Dr. G. Paavai Anand	Dr. M. Durgadevi	Dr. S. Prasanna Devi	Dr. C . Gomathy	Dr. C. V. Jayakumar
Organizing Secretary	Organizing Secretary	Organizing Secretary	HoD/CSE	VP Academics & Placements	Dean (CET) - VDP

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956
VADAPALANI

International Research Conference on IoT, Cloud and Data Science (IRCICD'22)

INSTITUTION'S INNOVATION COUNCIL
(Ministry of HRD Initiative)

Certificate of Participation

This is to certify that

Dr./Mr./Mrs./Ms. Harsita R

of SRM INSTITUTE OF SCIENCE AND TECHNOLOGY , VADAPALANI , CHENNAI

has participated and presented a **paper titled**

DETECTING FAKE JOB POSTING USING ML CLASSIFICATIONS & ENSEMBLE MODEL

in the Virtual International Research Conference on IoT, Cloud and Data Science (IRCICD'22) Organised by the Department of Computer Science and Engineering, SRM Institute of Science and Technology, Vadapalani, Chennai, Tamil Nadu, India held on 6th and 7th May 2022.

Dr. Golda Dilip	Dr. G. Paavai Anand	Dr. M. Durgadevi	Dr. S. Prasanna Devi	Dr. C . Gomathy	Dr. C. V. Jayakumar
Organizing Secretary	Organizing Secretary	Organizing Secretary	HoD/CSE	VP Academics & Placements	Dean (CET) - VDP

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956
VADAPALANI

International Research Conference on IoT, Cloud and Data Science (IRCICD'22)

INSTITUTION'S INNOVATION COUNCIL
(Ministry of HRD Initiative)

Certificate of Participation

This is to certify that

Dr./Mr./Mrs./Ms. Aadharsh K Praveen

of SRM INSTITUTE OF SCIENCE AND TECHNOLOGY , VADAPALANI , CHENNAI

has participated and presented a **paper titled**

DETECTING FAKE JOB POSTING USING ML CLASSIFICATIONS & ENSEMBLE MODEL

in the Virtual International Research Conference on IoT, Cloud and Data Science (IRCICD'22) Organised by the Department of Computer Science and Engineering, SRM Institute of Science and Technology, Vadapalani, Chennai, Tamil Nadu, India held on 6th and 7th May 2022.

Dr. Golda Dilip	Dr. G. Paavai Anand	Dr. M. Durgadevi	Dr. S. Prasanna Devi	Dr. C . Gomathy	Dr. C. V. Jayakumar
Organizing Secretary	Organizing Secretary	Organizing Secretary	HoD/CSE	VP Academics & Placements	Dean (CET) - VDP

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956
VADAPALANI

International Research Conference on IoT, Cloud and Data Science (IRCICD'22)

INSTITUTION'S INNOVATION COUNCIL
(Ministry of HRD Initiative)

Certificate of Participation

This is to certify that

Dr./Mr./Mrs./Ms. Niveditha S

of SRM INSTITUTE OF SCIENCE AND TECHNOLOGY , VADAPALANI , CHENNAI

has participated and presented a **paper titled**

DETECTING FAKE JOB POSTING USING ML CLASSIFICATIONS & ENSEMBLE MODEL

in the Virtual International Research Conference on IoT, Cloud and Data Science (IRCICD'22) Organised by the Department of Computer Science and Engineering, SRM Institute of Science and Technology, Vadapalani, Chennai, Tamil Nadu, India held on 6th and 7th May 2022.

Dr. Golda Dilip	Dr. G. Paavai Anand	Dr. M. Durgadevi	Dr. S. Prasanna Devi	Dr. C . Gomathy	Dr. C. V. Jayakumar
Organizing Secretary	Organizing Secretary	Organizing Secretary	HoD/CSE	VP Academics & Placements	Dean (CET) - VDP